**Aggregate View Obeject Detection network**

1. 实验介绍：

AVOD: an Aggregate View Obeject Detection network 一种多视图目标检测网络，来自于论文：Joint 3D Proposal Generation and Object Detection from View Aggregation。一种用于自动驾驶场景的聚合视图对象检测网络，所提出的神经网络架构使用LIDAR点云和RGB图像来生成。该网络由两个子网共享的特征：区域提议网络（RPN）和第二级检测器网络。所提出的RPN使用能够在高分辨率特征图上执行多模态特征融合的新颖架构，以为道路场景中的多个对象类生成可靠的3D对象提议。使用这些提议，第二阶段检测网络执行精确定向的3D边界框回归和类别分类，以预测3D空间中对象的范围，方向和分类。

代码：https：//github.com/kujason/avod

1. 实验原理

AVOD的结构如图2-1所示。

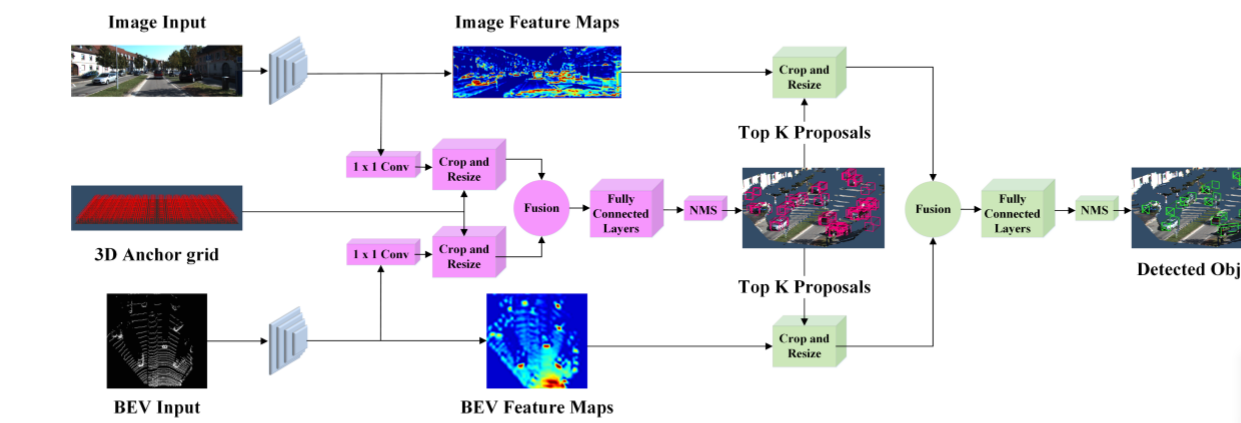


图2-1 AVOD的结构

1.Generating Feature Maps from Point Clouds and Images

从0.1米分辨率的点云的体素网格表示生成六通道BEV图。BEV图的前5个通道使用每个网格单元中的最大点高度进行编码，从沿Z轴的[0,2.5]米之间的5个相等划分生成，第六个BEV通道包含每个单元计算的点密度信息。

2. The Feature Extractor

RPN体系结构使用两个相同的特征提取器体系结构，图像输入和LIDAR输入各一个。全分辨率特征提取器如图2-2所示，由两段组成：编码器和解码器。编码器：对VGG-16做了一些修改，主要是将其通道数减半，并在conv4层裁剪网络。因此，编码器将一个M\*N\*D的图像或者鸟瞰图作为输入，并输出（M/8）×(M/8)×D\* 的特征图F。（行人一般是0.8×0.6m，在鸟瞰图中占8×6个像素（分辨率为0.1m）。经过编码器进行8倍下采样后，在输出的特征图中只占不到一个像素。）解码器：由FPN启发，设计了自底向上的解码器（bottom-up）将encoder输出的特征图上采样恢复至原始输入尺寸；通过conv-transpose（解卷积），将两个编码器输出的相关联的特征图级联，然后通过一个3×3卷积将两者融合。通过encoder和decoder最后得到的特征图具由较高的分辨率和代表性，并且是由RPN和the second stage detection network 共享的。

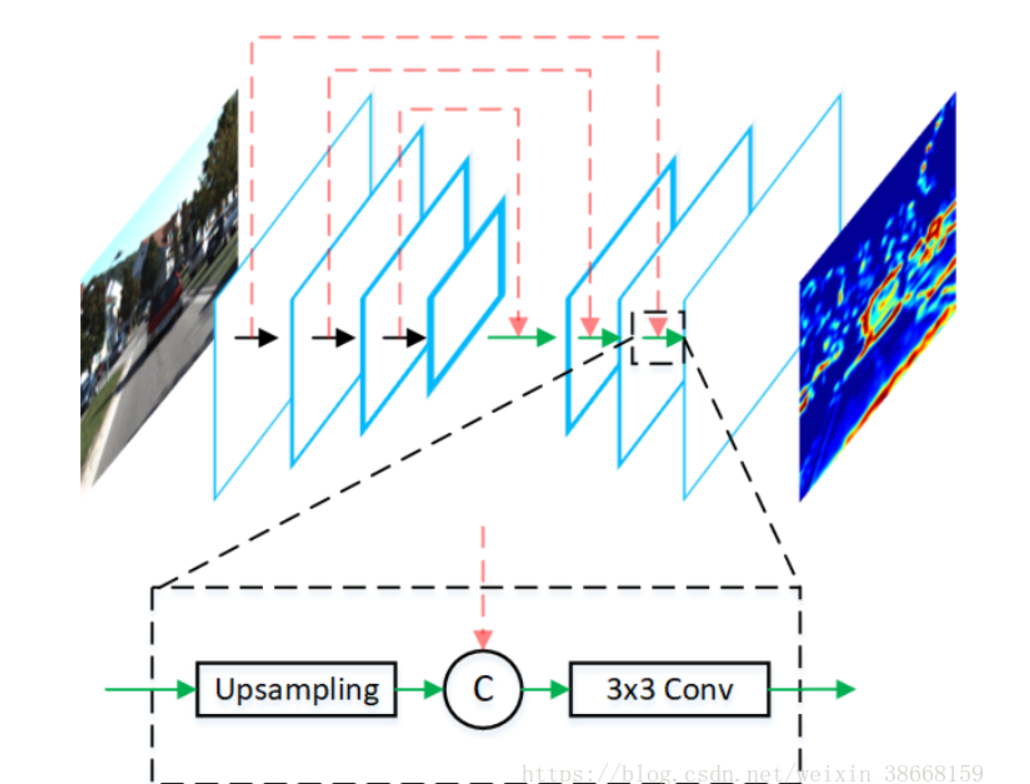


图2-2 特征提取器的结构

3. Multimodal Fusion Region Proposal Network

RPN回归了一组先前3D框与地面实况之间的差异。 这些先前的框被称为锚，轴对齐的边界框编码来编码，通过质心（tx，ty，tz）和轴对齐尺寸（dx，dy，dz）来参数化。为了生成3D锚网格，在BEV中以0.5米的间隔对（tx，ty）对进行采样，而tz基于传感器在地平面上方的高度来确定。 通过聚类每个类的训练样本来确定锚的尺寸。 BEV中没有3D点的锚点通过积分图像被有效地移除，导致每帧80-100K非空锚点。通过1×1卷积层降低维度：在某些情况下，区域提议网络需要在GPU内存中保存100K锚点的特征裁剪。两个大小为256的完全连接层的任务特定分支，使用融合特征裁剪来回归轴对齐的对象提议框并输出对象/背景“对象”分数。通过计算（Δtx，Δty，Δtz，Δdx，Δdy，Δdz），锚点和地面真实边界框之间的质心和尺寸的差异来执行3D框回归，通过计算锚和地面实况边界框之间的BEV中的2D IoU来确定背景锚。

4. Second Stage Detection Network

3D Bounding Box Encoding：

8角编码没有考虑3D边界框的物理约束，因为边界框的顶角被迫与底部的顶角对齐。为了减少冗余并保持这些物理约束，我们建议使用四个角和两个高度值对边界框进行编码，这两个高度值表示从传感器高度确定的地平面的顶部和底部拐角偏移。

Explicit Orientation Vector Regression

使用回归的方向向量来解决从采用的四角表示中的边界框方向估计的模糊性，因为这实验上发现比直接使用回归方向更准确。首先，提取边界框的四个可能方向，然后选择最接近显式回归方向向量的方向。

Detection：

与RPN类似，多视图检测网络的输入是通过将提议投影到两个输入视图中而生成的特征作物。与RPN类似，多视图检测网络的输入是通过将提议投影到两个输入视图中而生成的特征作物。

1. 实验过程

系统/软件：Ubuntu16.04、python3.5、TensorFlow1.3.0、miniconda

GPU：Titan Xp

1. 数据集准备：

数据集比较大（大约40G左右），而且很综合，各种格式的都有，不仅用了激光数据，而且用了图像数据进行训练**。**[Kitti Object Detection Dataset](http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d)、还有下载train.txt、val.txt、trainval.txt、planes，放到home根目录，如下图3-1所示。

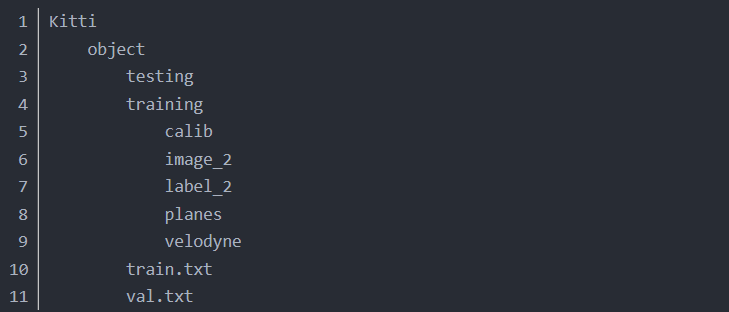


图3-1 数据集

1. 配置环境

在服务器中Ubuntu16.04搭建虚拟环境conda，在虚拟环境中搭建python3.5、cuda 8.0、 TensorFlow1.3.0。

1. 实验运行

安装python依赖：

cd avod

pip3 install -r requirements.txt

pip3 install tensorflow-gpu==1.3.0

添加环境变量：

add2virtualenv .

add2virtualenv wavedata

# For nonvirtualenv users

export PYTHONPATH=$PYTHONPATH:'/path/to/avod'

export PYTHONPATH=$PYTHONPATH:'/path/to/avod/wavedata'

完成图像积分：

sh scripts/install/build\_integral\_image\_lib.bash

Protobufs来配置模型：

sh avod/protos/run\_protoc.sh

mini-batch生成：

python scripts/preprocessing/gen\_mini\_batches.py

进行训练：

Pythonavod/experiments/run\_training.py --pipeline\_config=avod/configs/avod\_cars\_example.config

在TITAN XP上训练了16个小时，然后进行评估：

python avod/experiments/run\_evaluation.py --pipeline\_config=avod/configs/avod\_cars\_example.config --device='0' --data\_split='val'

训练时的图片如3-2，评估过程如图3-3所示

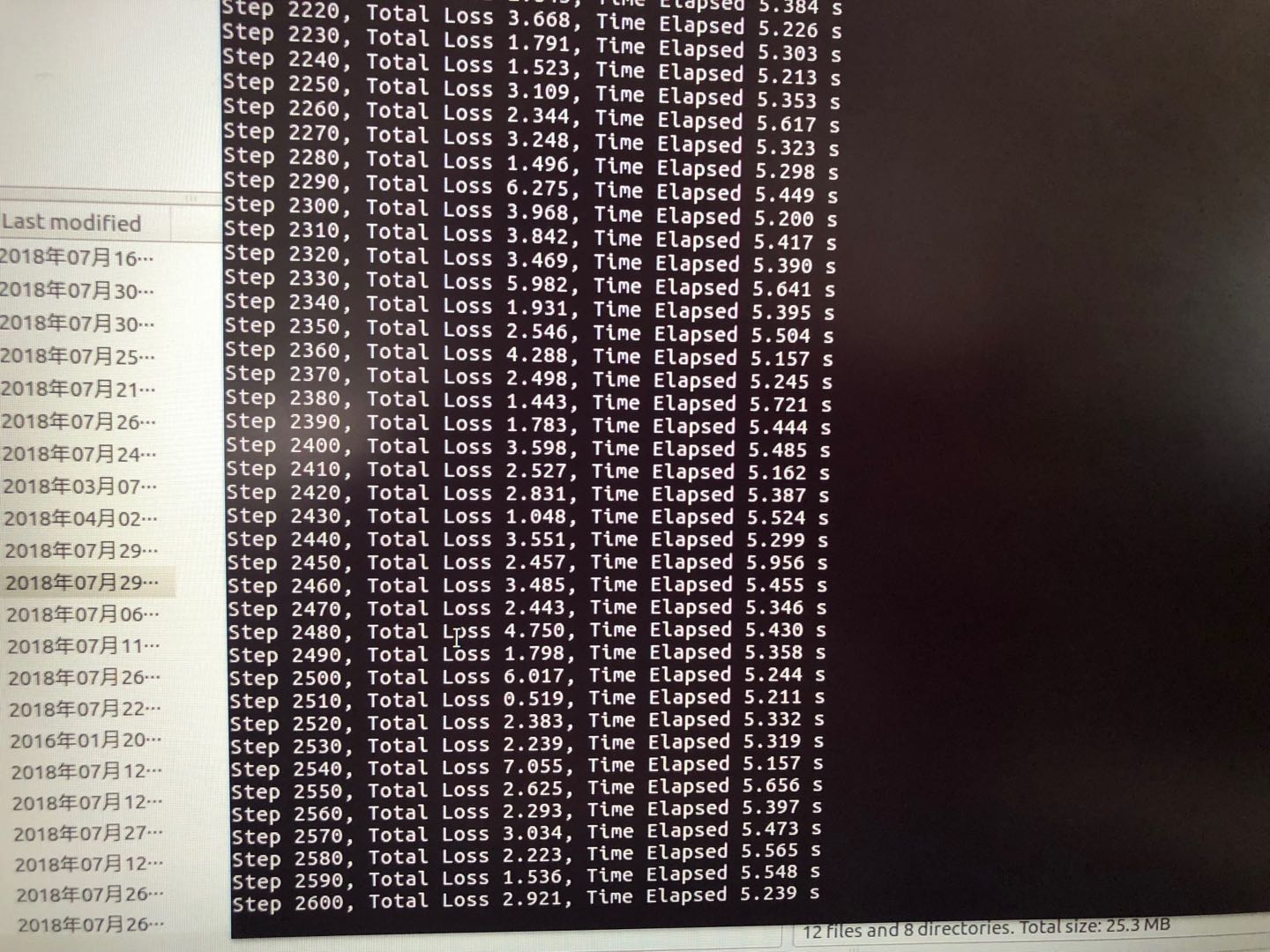


图3-2 训练过程

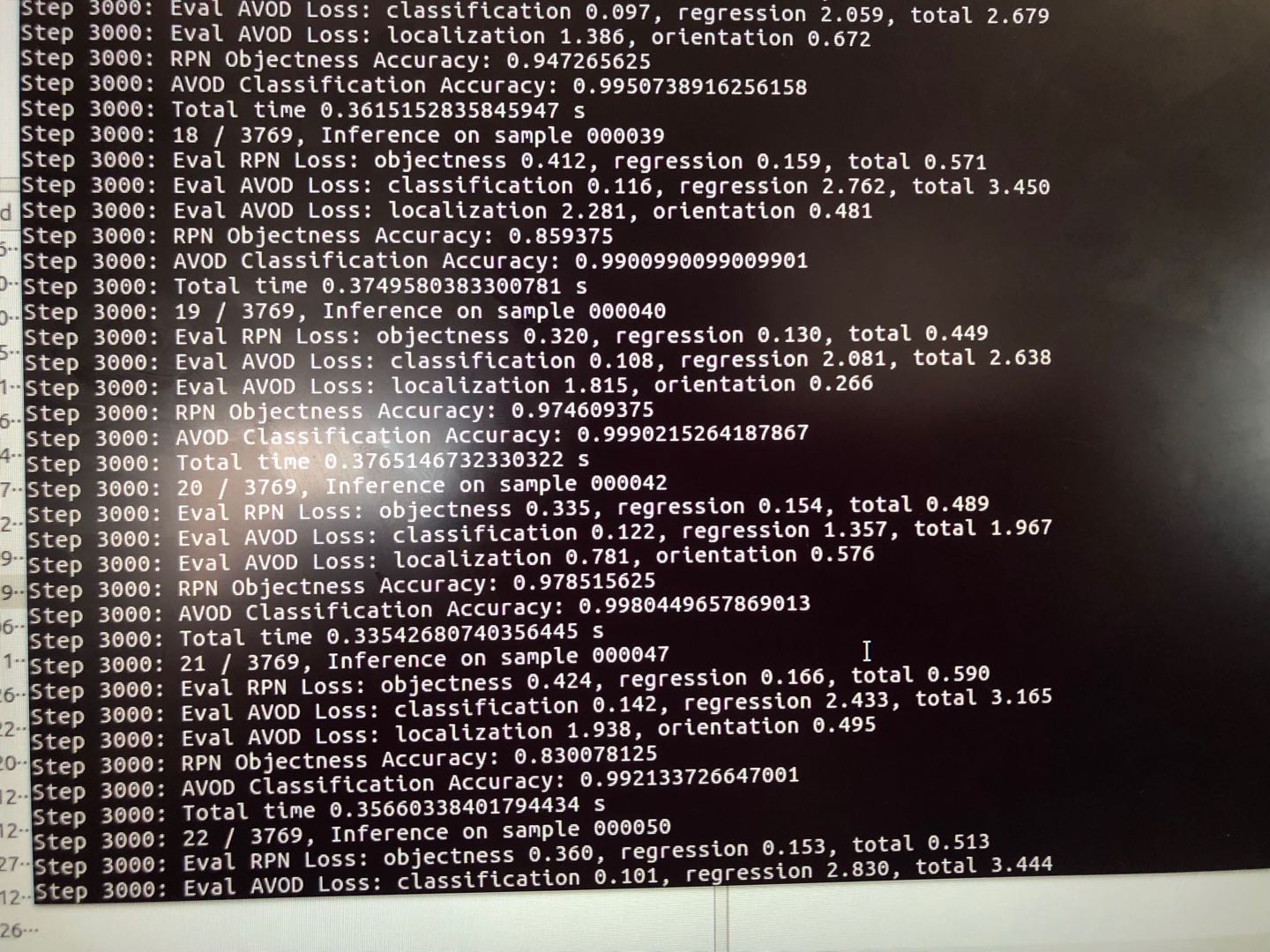


图3-3 评估过程

1. 实验结果

在实验结果，如图4-1的显示中可以看出可以达到论文中的结果水平如图4-2，本次实验目前只对车辆进行了检测。

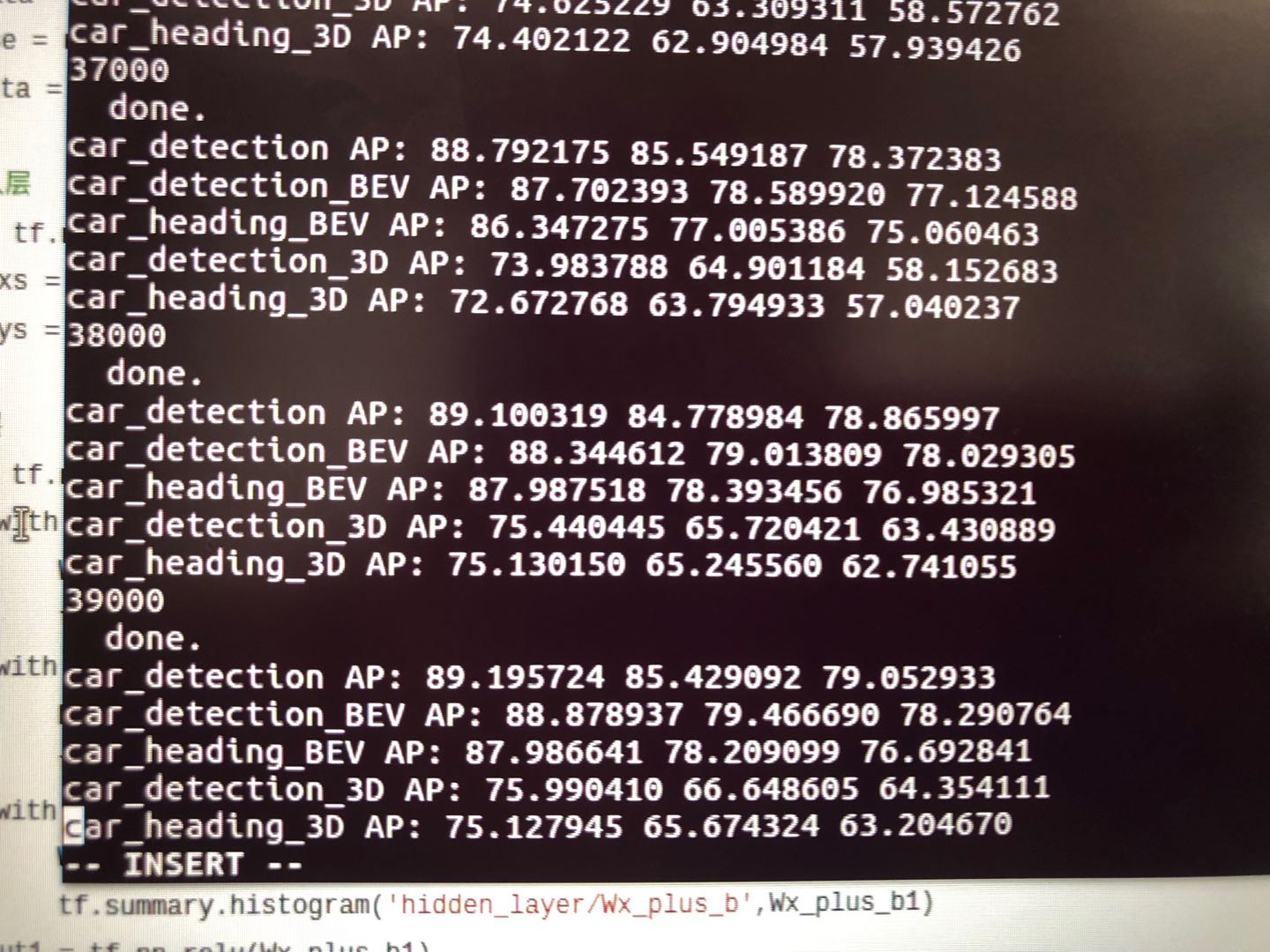


图4-1 实验结果

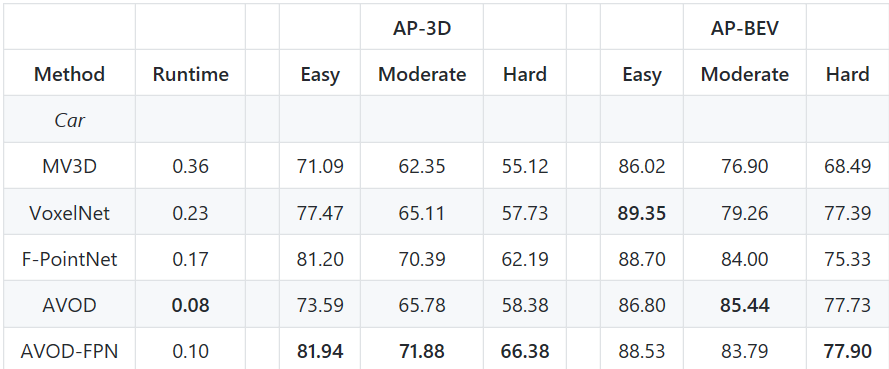


图4-2 论文中的结果

show\_predictions\_2d.py文件运行效果是图像中车辆的检测结果，包括分数，类别以及包围框，实验一共生成7480张结果图，其中4张如下所示。

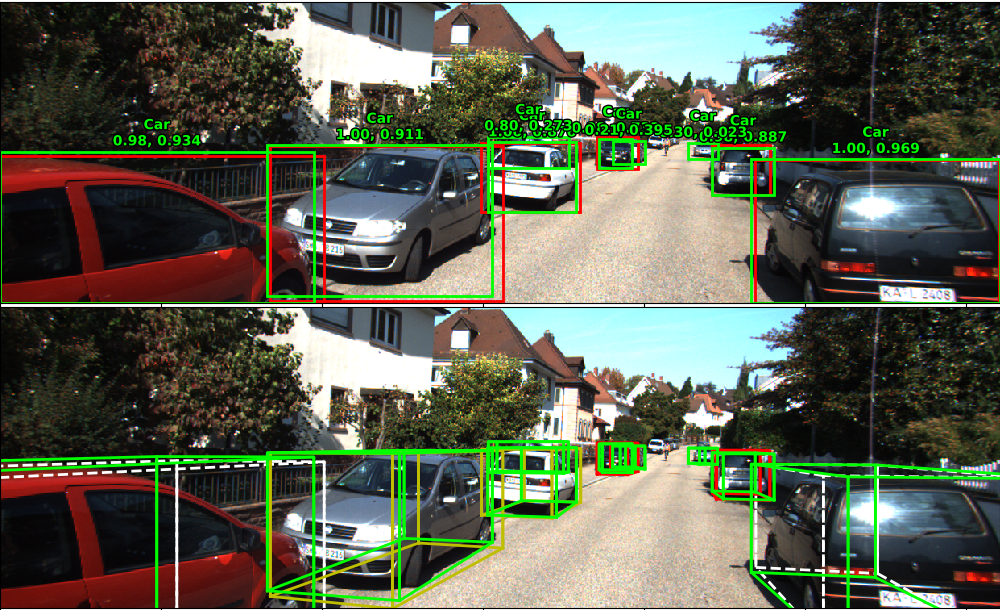


图4-3



图4-4



图4-5

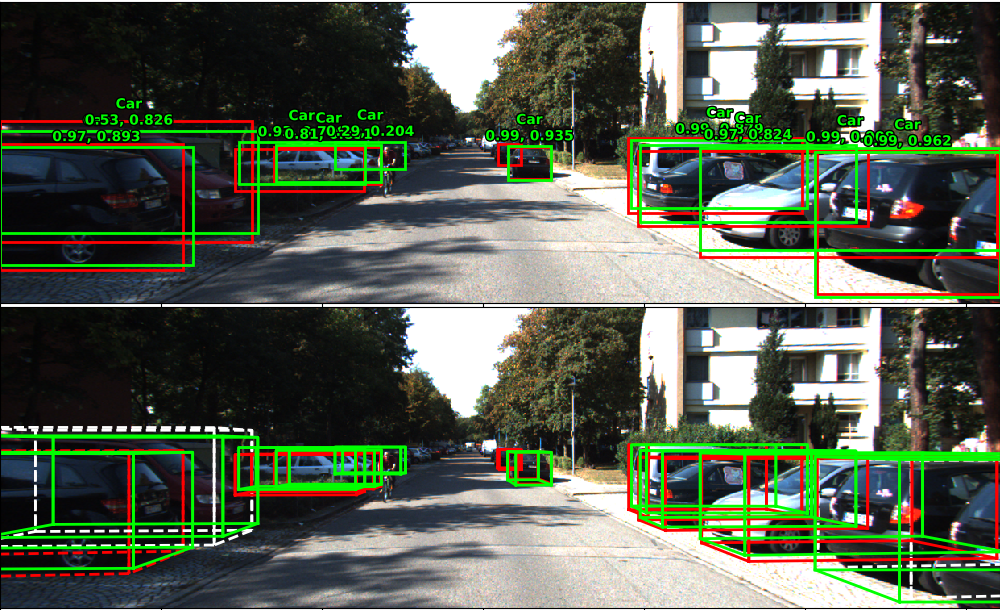


图4-6

kitti\_bev\_vis.py运行效果，这里会有一个图像数据，然后会有6张不同稀疏程度的64线雷达数据，绿色标示的就是检测到的车辆，这个也很容易在图像数据中发现右前方确实有一辆车。每一次执行这一步，场景都会有所不同。其中，每一个图像场景对应6张鸟瞰图，结果如面图如下所示。

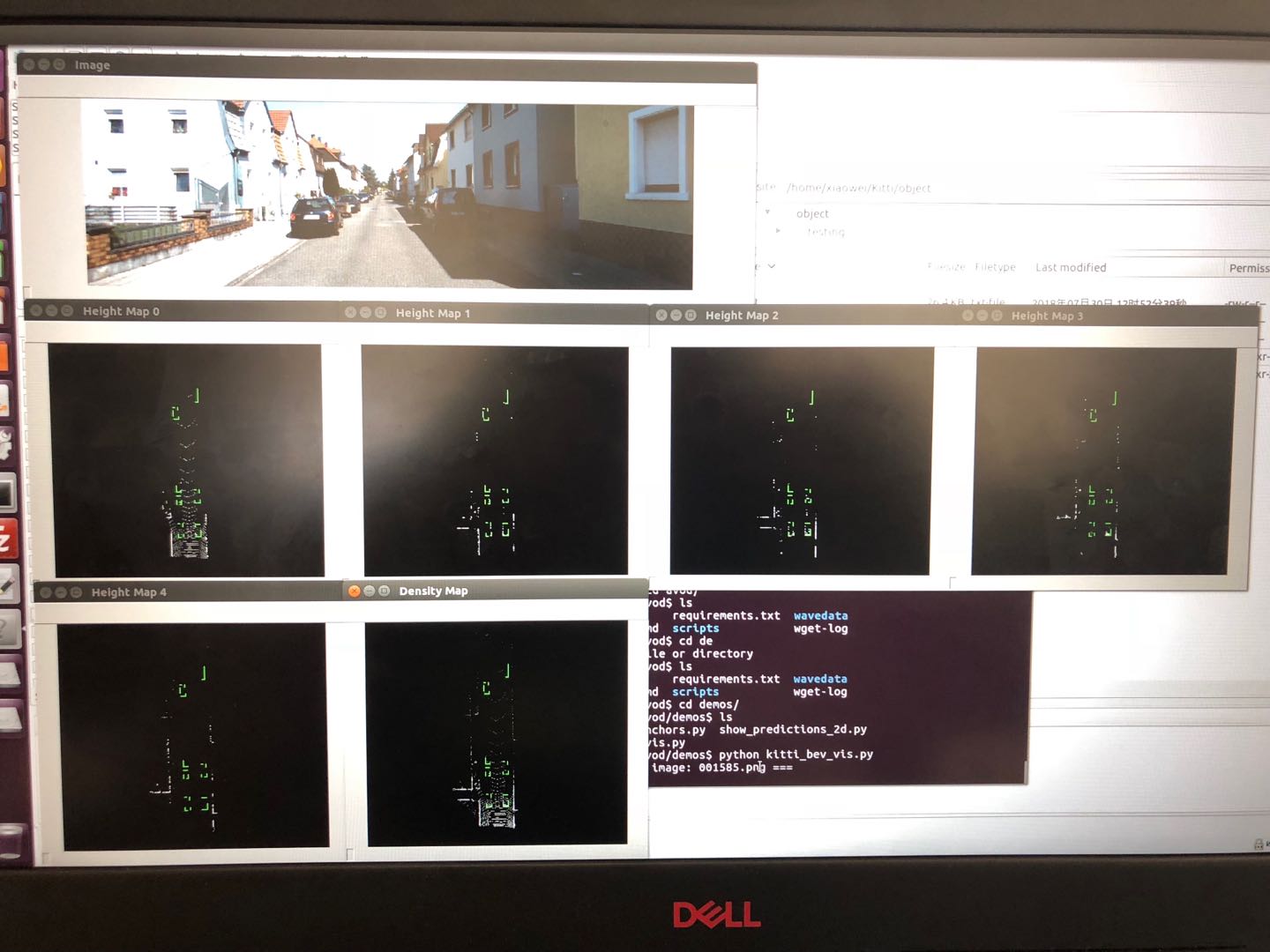


图4-7

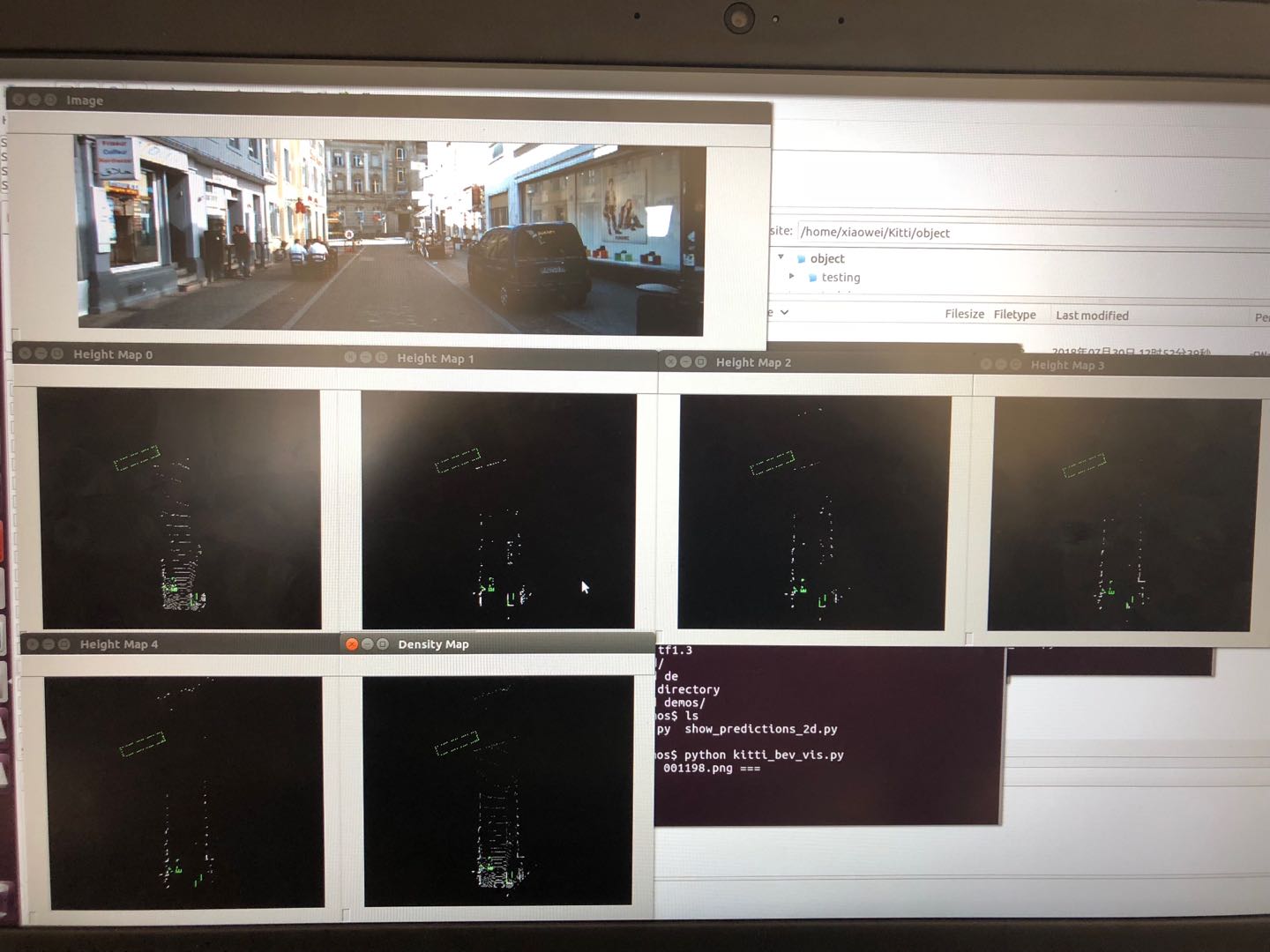


图4-8



图4-9 上图对应的放大图1

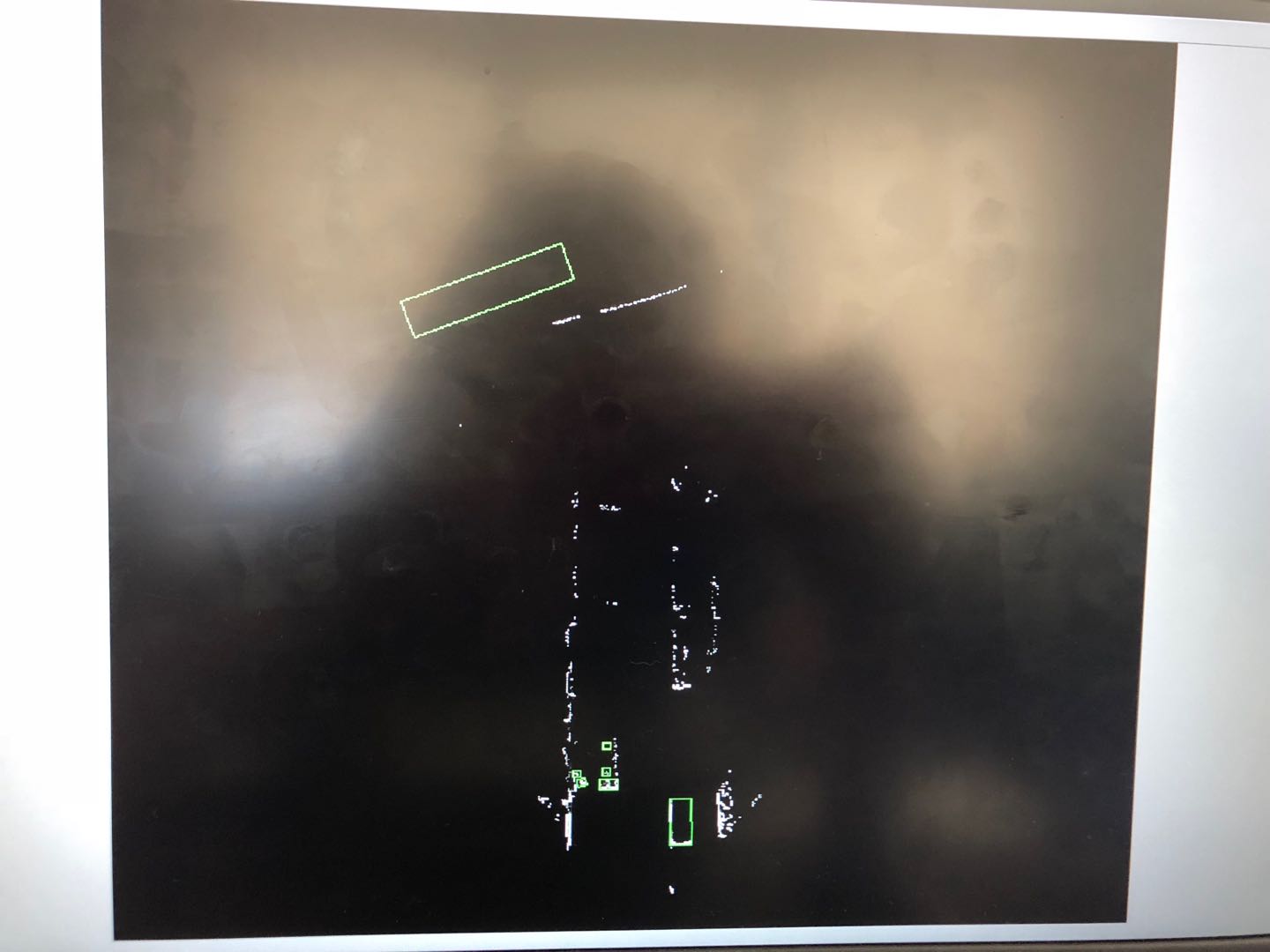


图4-10 上图对应的放大图2

小组成员：

姓名：鲁小伟 学号：2017Z8009061031

姓名：刘美月 学号：2017Z8009061080

姓名：尚旭鹏 学号：2016M8009073133

姓名：庞博 学号：2017Z8009061089

姓名：徐进 学号：2017Z8009061047