0-前言

为什么要这件事

无人驾驶行业涉及范围很广，应用到的往往是前言技术，资料又比较零散，少有成体系的讲理论与实战。我将要做的事情就是通过仿真的方式，带你通过可视的方法理解无人驾驶。

1-目标

项目的目标是给想要了解，转行无人驾驶或者，想要从事无人驾驶某个专项深耕的人，简历一个无人驾驶的知识星球。建立一个无人驾驶领域的业务体系，搭建出一套可debug的仿真环境，实战学习相关算法，提高个人能力提高专项技能。

2-方法

项目将以百度开源apollo项目为项目的开端，带你如何搭建一套可debug的环境，从实践入手，摆脱纯理论阶段。

一点点的通过仿真的方式，了解规控算法，感知算法等。一点点的深耕其实现原理和业务设计理论（业务上为什么要这么做）。

3-Apollo环境搭建

如果你还你点不了解无人驾驶，没关系。我们先搭建一套apollo环境，从实践的角度，跑起来看看怎末玩。

3.1-CPU版

3.1.1前置依赖硬件条件

特点是cpu要新,内存和存储要大

> |CPU:Intel 11代酷睿i9 11900（八核十六线程，最大睿频5.2GHz）
>
> |内存：威刚DDR4 32G 3200（可根据需求扩展至128G）
>
> |硬盘：西数黑盘500G

3.2-GPU版

> |CPU:Intel 11代酷睿i9 11900（八核十六线程，最大睿频5.2GHz）
>
> |内存：威刚DDR4 32G 3200（可根据需求扩展至128G）
>
> |硬盘：西数黑盘500G
>
> |显卡：NVIDIA GeForce RTX2060 12G

安装 NVIDIA 显卡驱动

可以通过界面的方式安装

随后，可以通过在终端中执行 `nvidia-smi` 命令来查看 NVIDIA 显卡工作是否正常（完成驱动安装后可能需要重启），正常情况下终端将显示下面的信息：

```
Sun Mar 27 10:35:07 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================
===|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0  On |                  N/A |
| 0%   37C    P8    20W / 184W |    553MiB / 12026MiB |      7%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================
===|
|    0   N/A  N/A      1113      G   /usr/lib/xorg/Xorg                215MiB |
|    0   N/A  N/A      1343      G   /usr/bin/gnome-shell               61MiB |
|    0   N/A  N/A      2819      G   /usr/lib/firefox/firefox          226MiB |
|    0   N/A  N/A      5156      G   ...40467804860571111 5,131072       8MiB |
|    0   N/A  N/A     24499      G   ...AAAAAAAAA= --shared-files       33MiB |
|    0   N/A  N/A     25945      G   /usr/lib/firefox/firefox            1MiB |
+-----------------------------------------------------------------------------+
```

安装 NVIDIA 容器工具包

如果是在物理机中安装的 Ubuntu，且机器配有 NVIDIA 显卡，在安装了驱动的前提下，还需要安装 NVIDIA 容器工具包以运行 Apollo Docker 镜像中的 CUDA：

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee
/etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get -y update
sudo apt-get install -y nvidia-docker2
```

前置依赖软件

- 安装 Ubuntu 18.04
- 安装 Git
- 安装 Docker 引擎

1.安装 Ubuntu 18.04

系统的安装参考网上教程;

2.安装 GIT

为什么要安装git,因为要从仓库上拉取代码;由于代码较大,所以使用ssh的方式最稳妥,不会中途断掉;所以要有一个github账号.

- ubuntu安装git
- 生成git ssh key
- 添加key 到GitHub上

参考链接

> https://www.cnblogs.com/zxlovenet/p/4571850.html

3.安装 Docker 引擎

安装Docker 19.03 及以上版本，在终端中直接执行下述命令即可完成 Docker 社区版的安装：

```
curl https://get.docker.com | sh
sudo systemctl start docker && sudo systemctl enable docker
```

重启 Docker 守护进程以使改动生效：

```
sudo systemctl restart docker
```

3.1.3克隆 Apollo 源码

推荐使用过 SSH 方式克隆 Apollo 源码仓库：

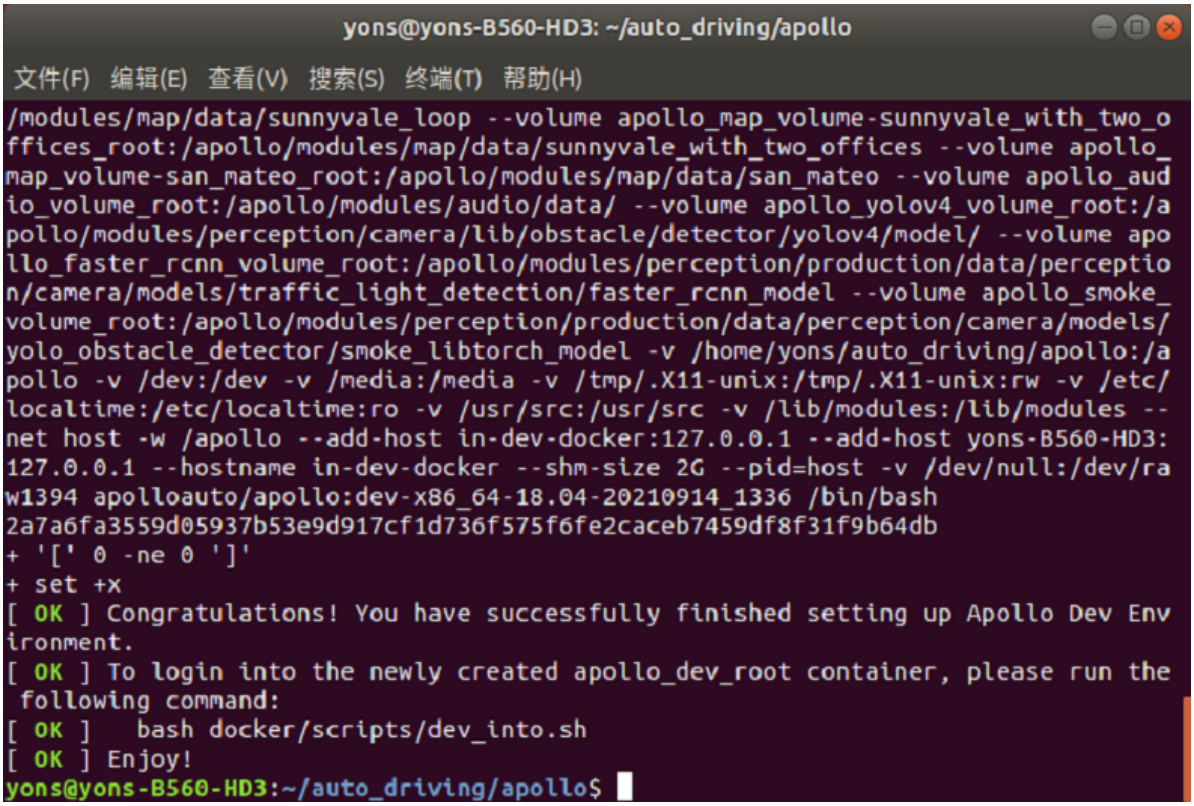
```
# 使用 SSH 的方式
git clone git@github.com:ApolloAuto/apollo.git
```

3.1.4启动 Apollo Docker 开发容器

进入到 Apollo 源码根目录，打开终端，执行下述命令以启动 Apollo Docker 开发容器：

```
sudo ./docker/scripts/dev_start.sh
```

不出意外得话，启动成功后将得到下面信息：

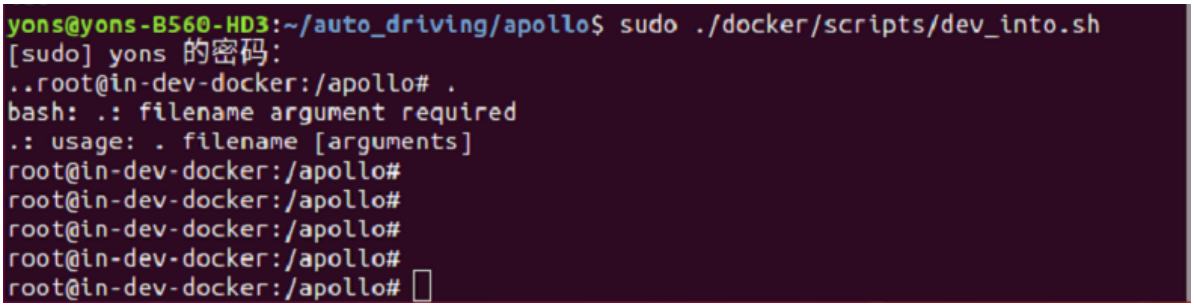


3.1.4进入 Apollo Docker 开发容器

启动 Apollo Docker 开发容器后，执行下述命令进入容器：

```
sudo ./docker/scripts/dev_into.sh
```

可以发现，进入容器后终端信息发生了相应变化，后面的操作都将在容器中进行：



若提示`[WARNING] nvidia-smi not found. CPU will be used.`请确认是否要用gpu编译，不是则忽略。若是，请检查NVIDIA 容器工具包 是否安装，重新执行`sudo ./docker/scripts/dev_start.sh`可解决。

3.1.5容器中构建 Apollo

进入 Apollo Docker 开发容器后，在容器终端中执行下述命令构建 Apollo：sudo ./apollo.sh build 自动适配用cpu编译还是gpu

```
sudo ./apollo.sh build
```

```
^C[yons@in-dev-docker:/apollo]$ sudo bash apollo.sh build_dbg
[INFO] Apollo Environment Settings:
[INFO]     APOLLO_ROOT_DIR: /apollo
[INFO]     APOLLO_CACHE_DIR: /apollo/.cache
[INFO]     APOLLO_IN_DOCKER: true
[INFO]     APOLLO_VERSION: master-2022-02-07-a4aa021bcd
[INFO]     DOCKER_IMG:
[INFO]     APOLLO_ENV:  STAGE=dev USE_ESD_CAN=false
[INFO]     USE_GPU: USE_GPU_HOST= USE_GPU_TARGET=1
[ OK ] Running GPU build on x86_64 platform.
[WARNING] ESD CAN library supplied by ESD Electronics doesn't exist.
[WARNING] If you need ESD CAN, please refer to:
[WARNING]   third_party/can_card_library/esd_can/README.md
[INFO] Build Overview:
[INFO]     USE_GPU: 1  [ 0 for CPU, 1 for GPU ]
[INFO]     Bazel Options: --config=gpu --config=dbg
[INFO]     Build Targets: //modules/... union //cyber/...
[INFO]     Disabled:      except //modules/drivers/canbus/can_client/esd/...
Starting local Bazel server and connecting to it...
(02:46:28) INFO: Invocation ID: 257608e3-2718-4b59-b771-24c22991926a
(02:46:28) INFO: Current date is 2022-03-27
(02:46:28) Loading: 0 packages loaded
```

用时739秒编译成功后如下 Enjoy!

```
(13:57:25) INFO: Elapsed time: 739.237s, Critical Path: 153.56s
(13:57:25) INFO: 9672 processes: 5761 internal, 3911 local.
(13:57:25) INFO: Build completed successfully, 9672 total actions
==============================================
[ OK ] Done building apollo. Enjoy!
==============================================
```

3.1.6运行 Apollo

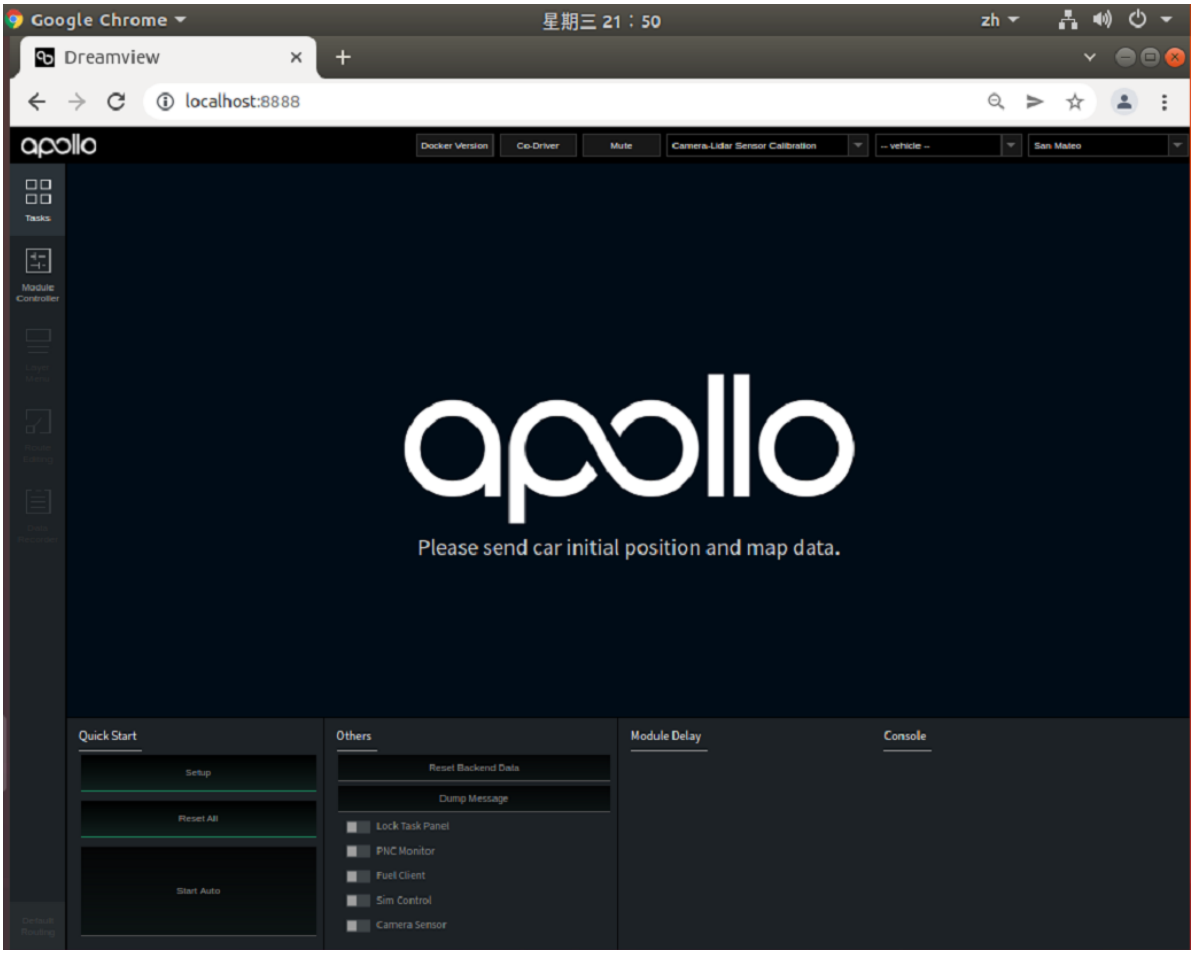
1 启动 Apoll

完成 Apollo 构建后，在容器终端中执行下述命令：

```
./scripts/bootstrap.sh start
```

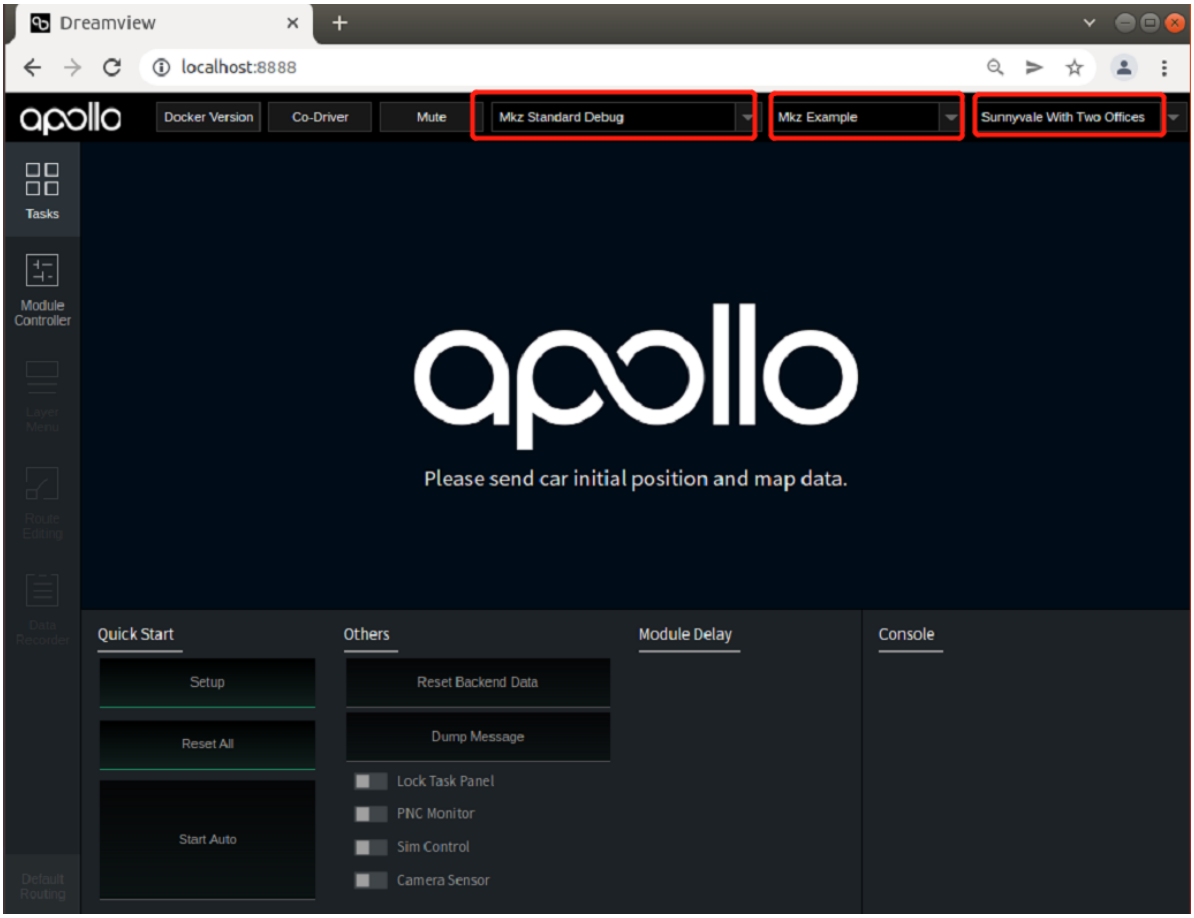
```
root@in-dev-docker:/apollo# ./scripts/bootstrap.sh start
[WARNING] nvidia-smi not found. CPU will be used.
[WARNING] nvidia-smi not found. CPU will be used.
[WARNING] nvidia-smi not found. CPU will be used.
[WARNING] nvidia-smi not found. CPU will be used.
nohup: appending output to 'nohup.out'
[ OK ] Launched module monitor.
[WARNING] nvidia-smi not found. CPU will be used.
[WARNING] nvidia-smi not found. CPU will be used.
nohup: appending output to 'nohup.out'
[ OK ] Launched module dreamview.
Dreamview is running at http://localhost:8888
root@in-dev-docker:/apollo#
```

上述命令会启动 DreamView 并使能模块监控机制，在浏览器中访问 http://localhost:8888 来显示 DreamView 界面：



2.选择驾驶模式和地图

在 DreamView 界面的对应下拉框中选择驾驶模式为“Mkz Example，选择地图为“Sunnyvale with Two Offices”：



3.回放 Demo 数据

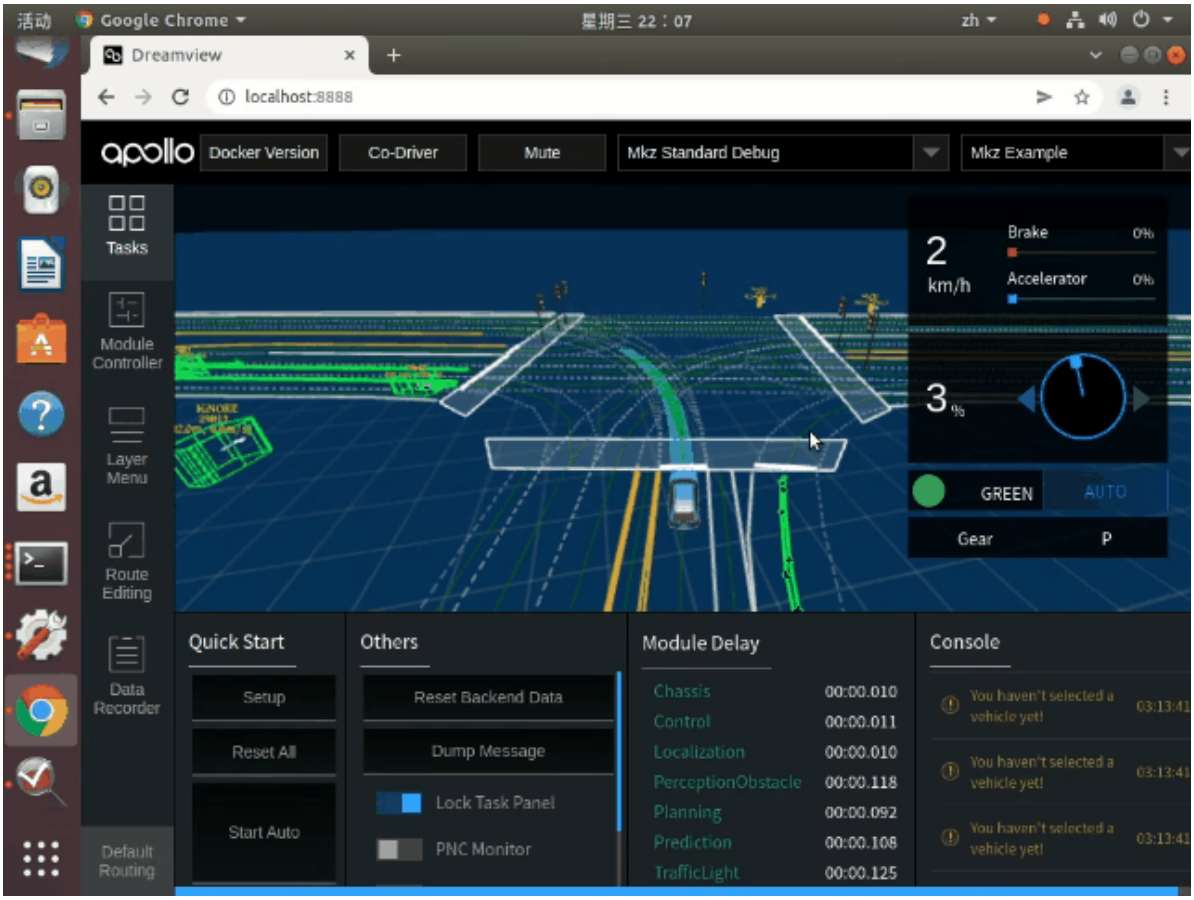
在容器终端中执行下述命令下载 demo 数据：

```
source cyber/setup.bash # 回放包需要设置下环境变量
cd docs/demo_guide/
python3 record_helper.py demo_3.5.record
```

由于网络原因，下载可能失败，可以点击这里手动下载并将数据放到 `apollo/docs/demo_guide/` 目录下。继续在容器终端中执行下述命令来播放数据，`-l` 表示循环播放（loop）：

```
cyber_recorder play -f demo_3.5.record -l
```

至此，DreamView 界面中将呈现出自车规划轨迹、他车预测轨迹、路网等各种信息：



参考

https://blog.shipengx.com/archives/e4b9c8ad.html

https://blog.csdn.net/weixin_45929038/article/details/120113008