

综合设计（一）

➤ 题目：

基于 AI 的车载智能终端设计与实现

➤ 背景：

摄像头是自动驾驶中常用的传感器，有着价格低廉、采集数据信息丰富和轻便等优点，基于视觉的感知方案对自动驾驶中避免碰撞和规划路径非常重要。本实验以车辆搭载单目摄像头实现碰撞提醒、车道偏移为应用场景，利用边缘计算平台进行处理，实现前车检测跟踪测距、车道线检测、实时显示等功能。



➤ 教学目标：

目标让学生掌握轻量型 AI 平台实现方法，掌握图像目标检测、测距、跟踪方法，车道线检测方法、掌握 QT 实时显示以及直播整体系统环境搭建方法。

➤ 目的、要求：

支持车道线检测、车辆检测、以及车辆测距识别；

RTSP 拉流解码；

基于霍夫变换的车道线检测+标记；

前方车辆检测+车辆距离识别；

前车距离太近时给予告警；

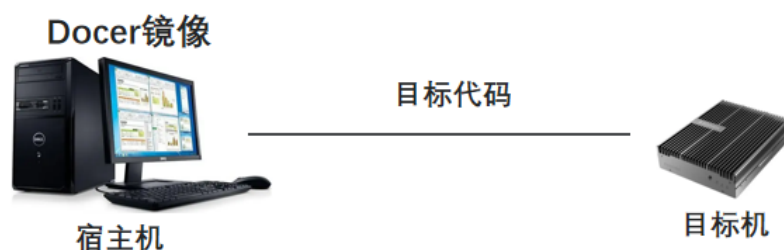
视频流分段实时存储；

QT 实时检测结果显示；

➤ 实验器材与开发环境：

项目所使用的软硬件设备配置见下表：

配置	参数
本地编译系统	Ubuntu18.04(虚拟环境)
推流系统	Windows 10（或 Linux）
CPU	Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz
边缘计算盒	AI 计算盒 SE5 SE50221
加速框架	Inference framework
主要工具包	sophonsdk_v3.0.0、QTlib、gcc-linaro-6.3.1、Yolov5s
深度学习框架	Pytorch
推流软件	EasyDarwin8.1.0、ffmpeg、VLC media player



➤ 实验步骤

1. 程序文件

项目由以下文件夹构成：

visual_perception_proj

- ├--bin
- ├--calib
- ├--docs
- ├--results
- ├--source
- └--video

其中，calib 是相机标定模块，docs 是环境部署和代码编译文档，results 中包含代码运行的结果，source 包含代码文件夹 code、QT 库 install.zip、配置参数 cameras.json、深度学习模型 yolov5s_1batch_fp32.bmodel、运行脚本 run_hdmi_yolo_line.sh，video 文件夹中是测试视频，results 中包含测试结果视频，bin 文件夹中包括可执行文件。

2. 基础环境搭建

2.1 虚拟环境搭建

我们需要一个宿主机，并在其上搭建交叉编译环境，能够编译出能够在目标机上运行的可执行文件。此处选择创建一个搭载 Ubuntu 系统的虚拟机作为宿主

机（如果你的系统本来就是 Ubuntu，那么可以跳过这一部分）。

首先在 VMware 官网下载虚拟机软件，此处使用的是 VMware Workstation 17 PRO。安装时若没有购买激活码，则可以临时选择试用。（VMware 官网：<https://www.vmware.com/cn/products/workstation-pro.html>）

下载所需的 Ubuntu 系统镜像，下载链接为（<https://developer.aliyun.com/mirror/>）。打开网站后点击“OS 镜像”：

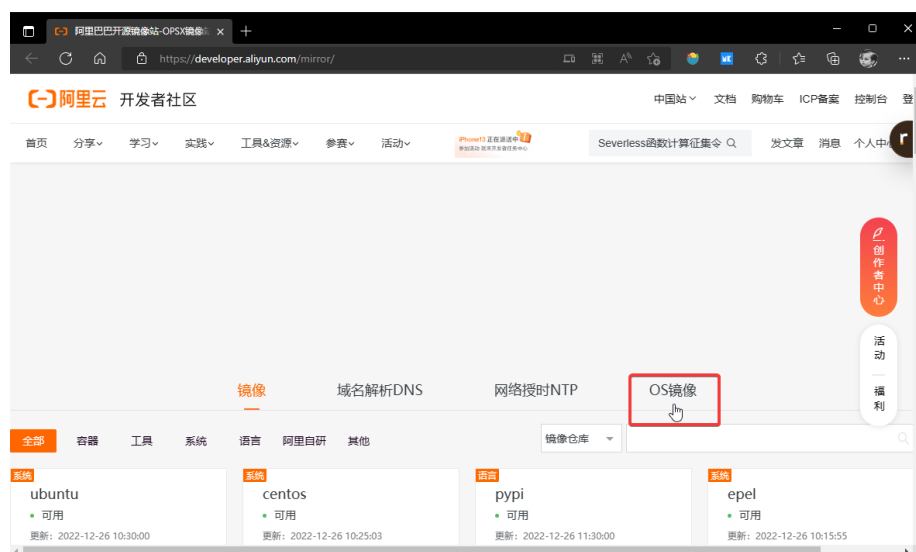


图 2.1-(1) 阿里云的镜像下载站

发行版选择“ubuntu”，版本选择“18.04.6(desktop-amd64)”，然后点击下载即可：

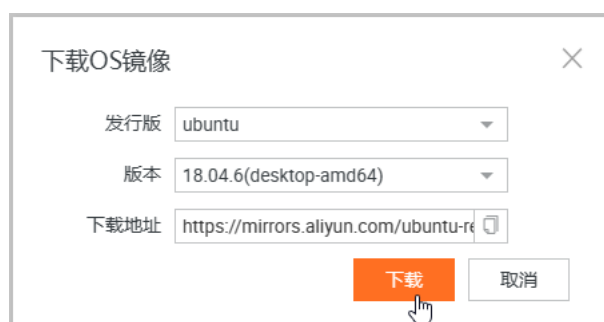


图 2.1-(2) 镜像选择

下载完成后，打开 VMware 虚拟机软件，点击主页上的“新建虚拟机”选项：

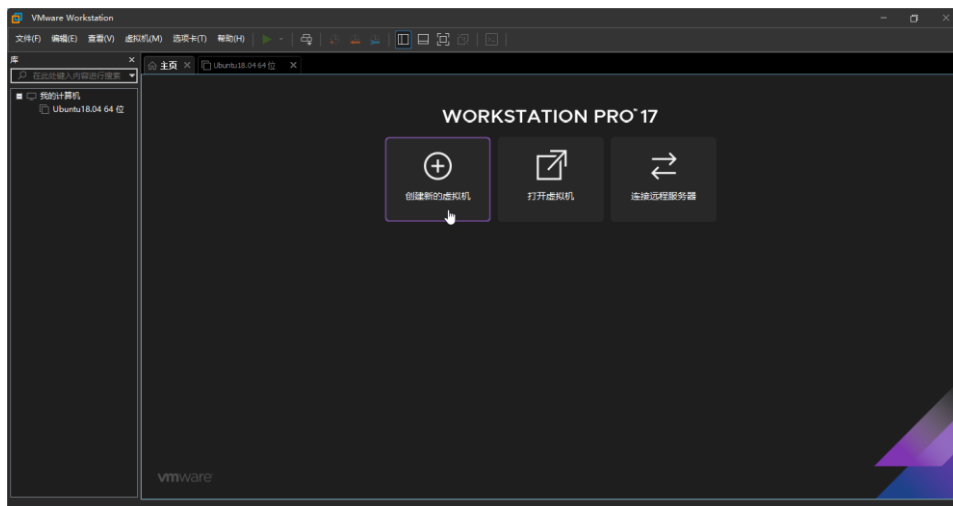


图 2.1-(3) VMware workstation 主界面

在新建虚拟机向导中选择“典型”，点击下一步：



图 2.1-(4) 选择配置方式

选择刚才下载下来的镜像，等待识别镜像类型后即可实现自动化的配置：

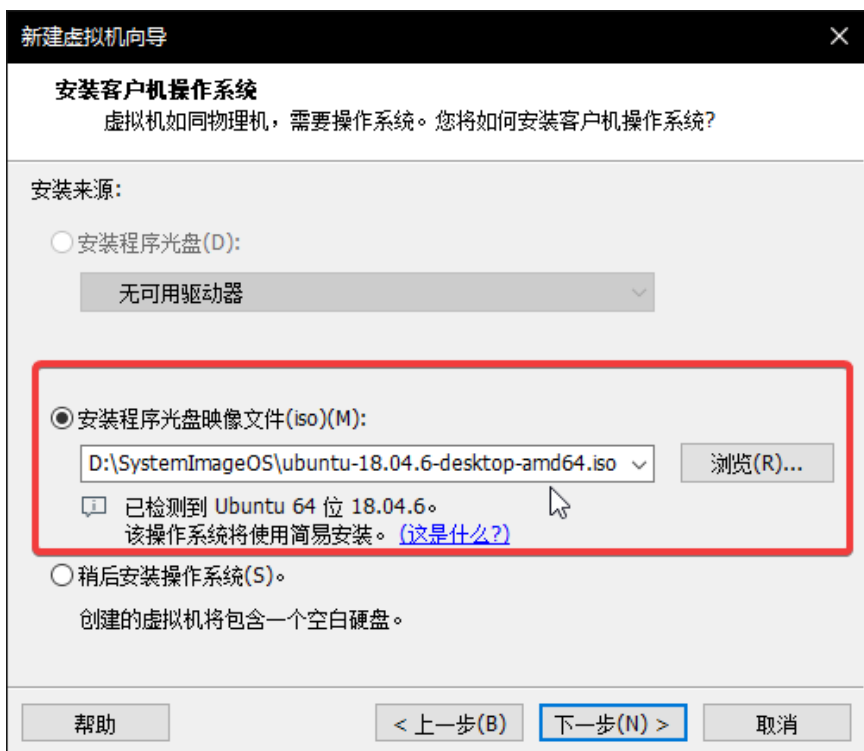


图 2.1-(5) 选择镜像路径

注意，这里的用户名不能使用“root”或“admin”，这两个用户名是系统内置的管理员账户，所以在自动化配置的时候将会出现错误。同时不建议将用户姓名设置为以上两个，避免混淆。



图 2.1-(6) 用户名及密码配置

选择虚拟系统的安装目录：

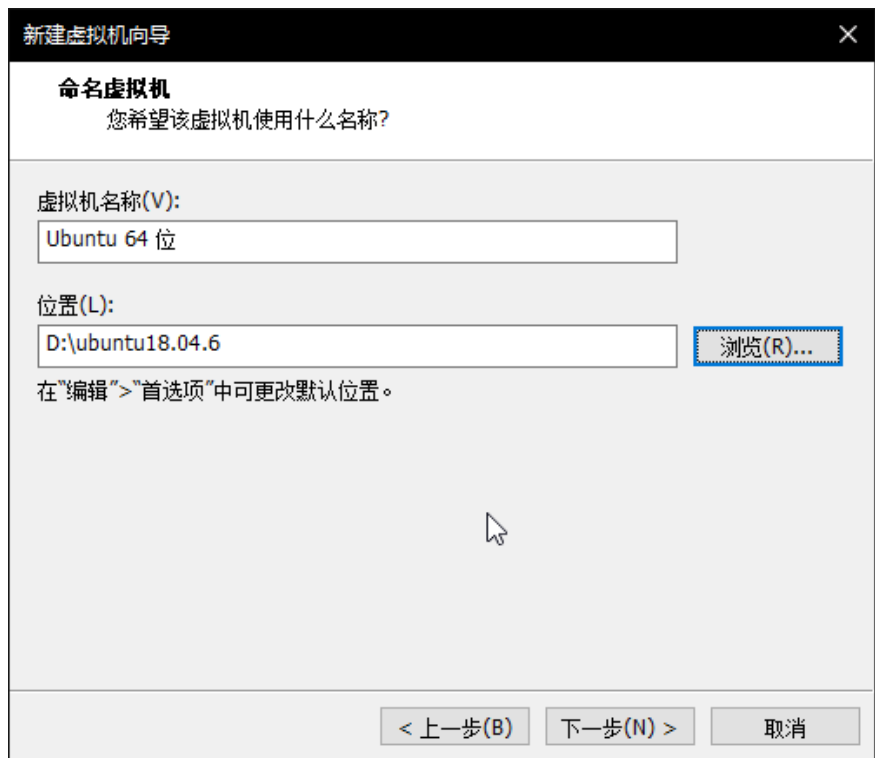


图 2.1-(7) 选择虚拟系统安装目录

这里分配存储空间时，如果有足够的剩余的存储空间，建议尽可能多分配一些。以免后续使用过程中出现存储空间不足的情况，调整为 50GB（建议大于 30GB）：

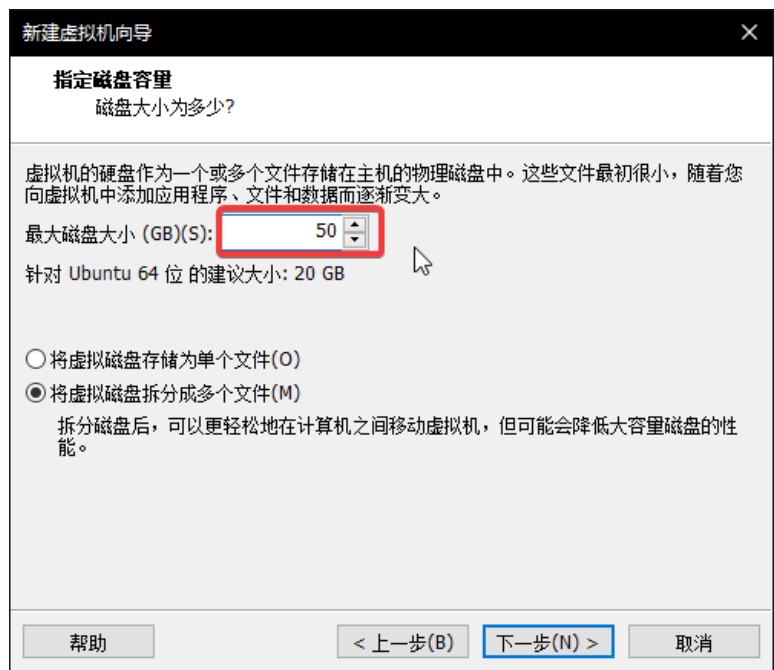


图 2.1-(8) 分配虚拟系统存储空间

确认配置后，点击完成，就开始自动安装虚拟环境了。

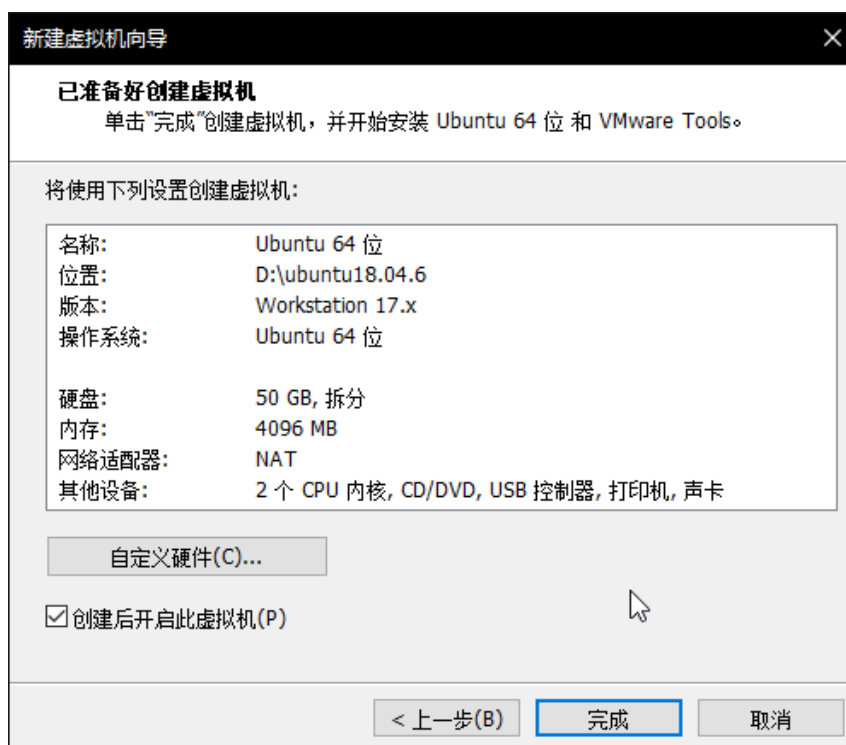


图 2.1-(9) 确认配置

虚拟机安装完成后，为了便于使用，我们还需要重新配置下载源。因为 Ubuntu 原生的服务器并不在中国境内，在后续配置编译环境下载依赖时会十分缓慢，且时常有下载出错的现象。好在国内有与之对应的镜像源，以方便中国大陆用户使用。下面介绍如何配置国内的镜像下载源。

首先点击 Ubuntu 系统左下角的“应用”：

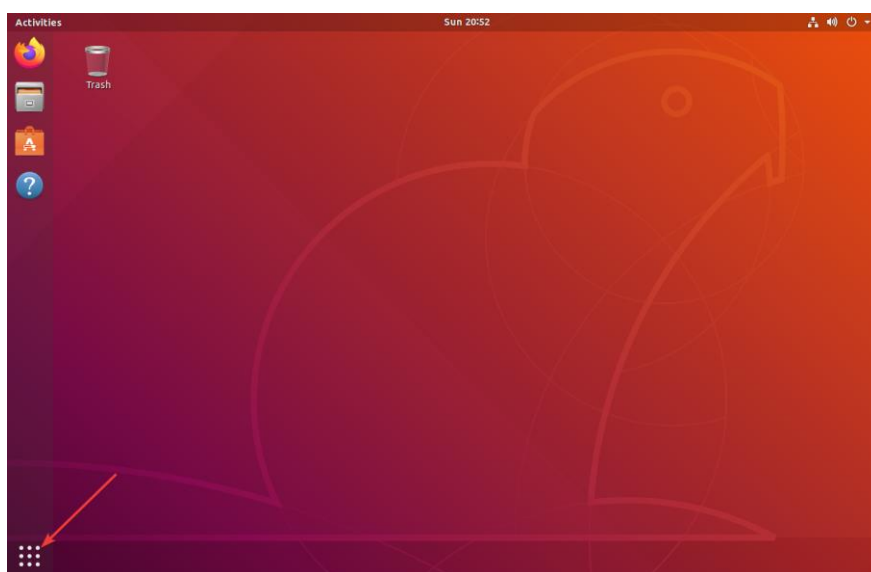


图 2.1-(10) “应用”按钮

然后选择“Software & Updates”：

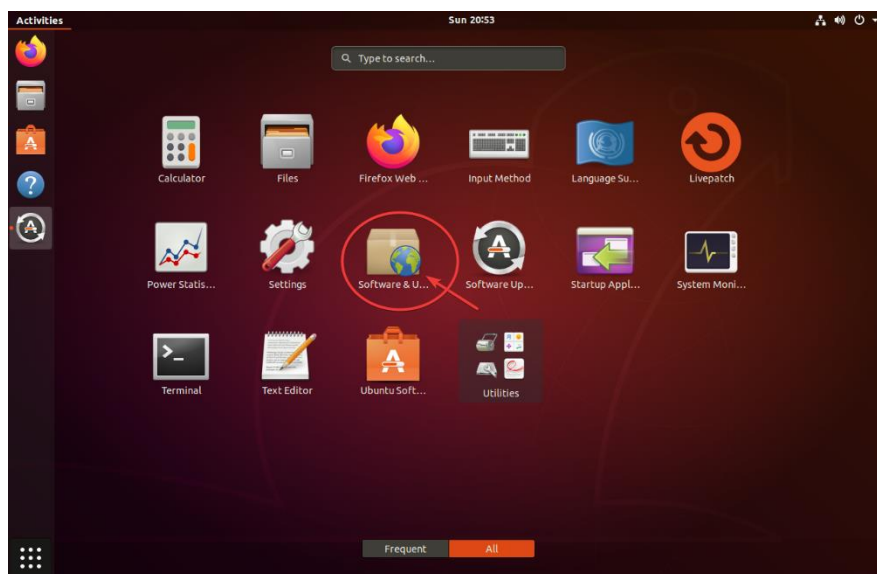


图 2.1-(11) Software&Updates

点击“Download from”：

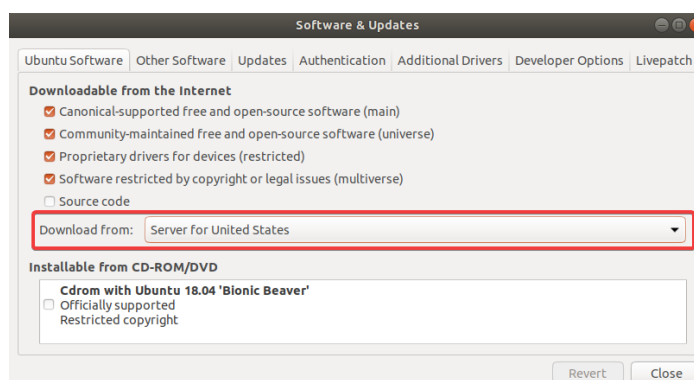


图 2.1-(12) 选择下载源

点击右上角的“Select Best Server”来让系统自动匹配速度最快的镜像源：

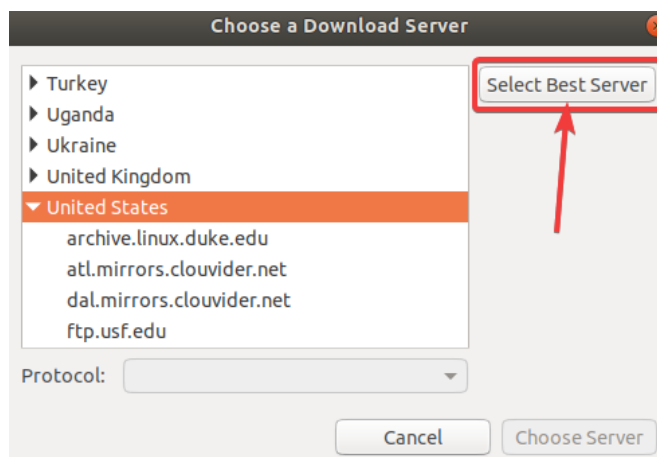


图 2.1-(13) 自动选择镜像源

等待加载完成后，点击右下角的“Close”，这时会弹出确认提示。点击“Reload”来重新加载以应用新选择的下载源：

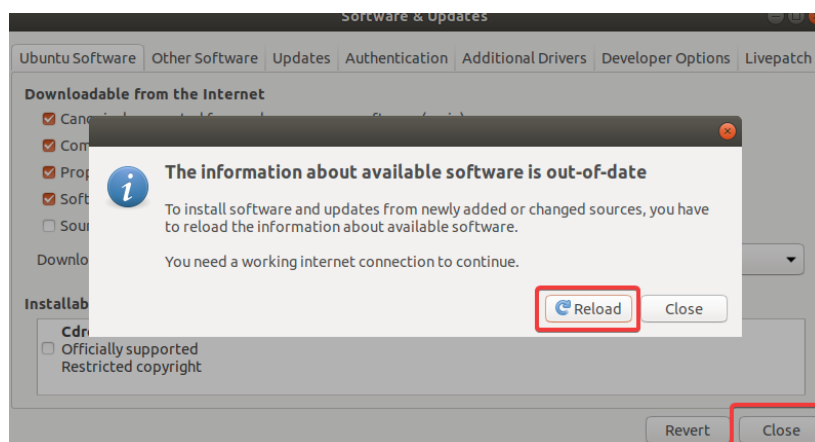


图 2.1-(14) 重新载入

然后等待载入完成即可：

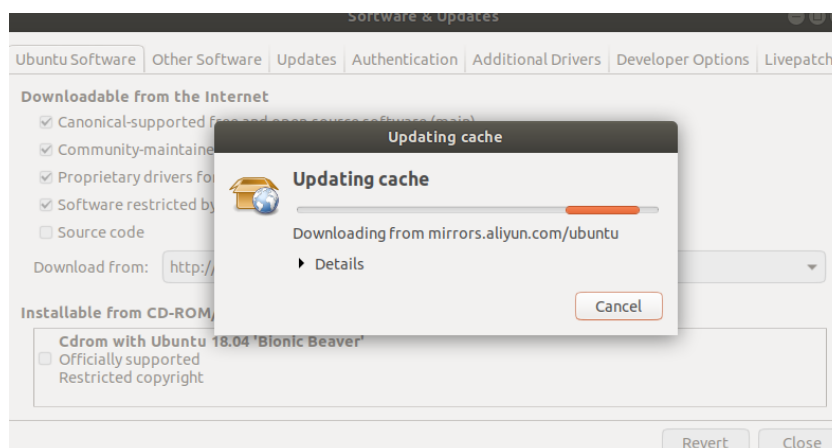


图 2.1-(15) 等待加载完成

除此之外，可以选择将系统语言配置为中文。

打开左下角的应用栏（如图 2.1-(10) “应用”按钮），选择设置：

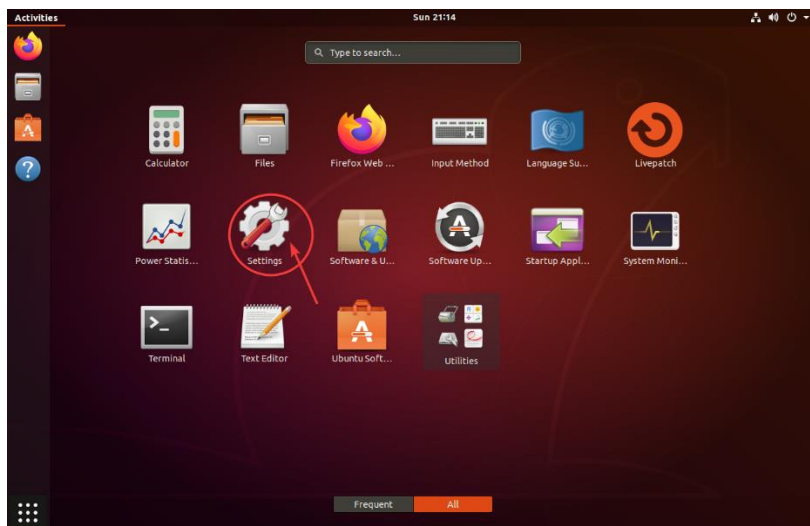


图 2.1-(16) “设置”按钮

选择左侧选项卡中的“Region & Language”，点击“Manage Install”：

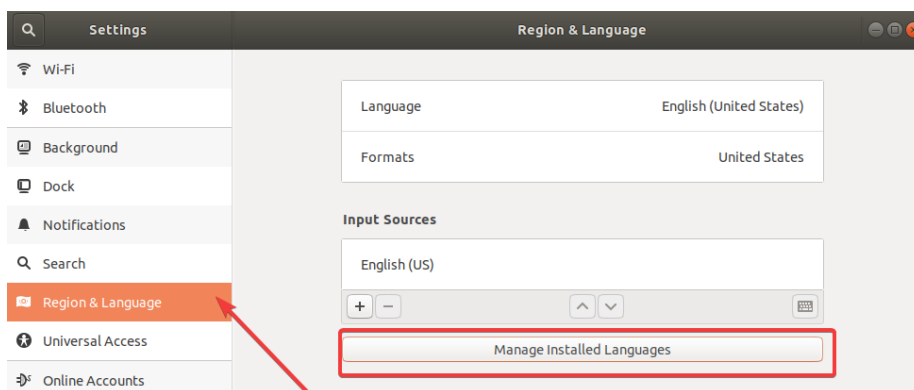


图 2.1-(17) 进入语言设置

由于此系统为全新安装的系统，所以还需要安装完整的语言支持。在弹出的窗口直接点击“Install”即可：

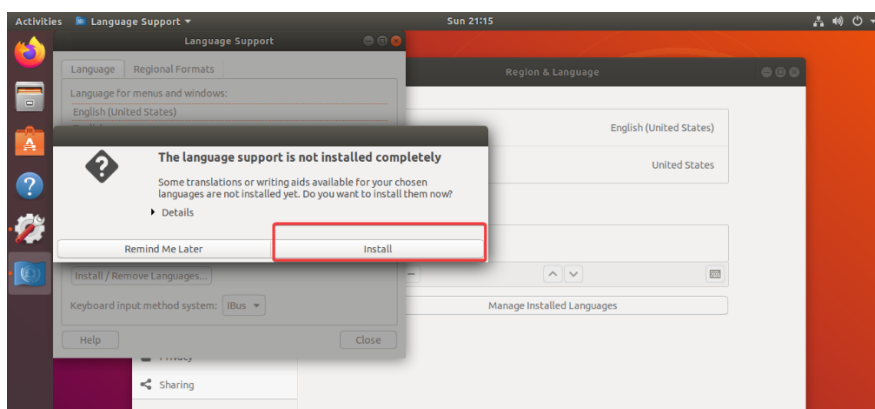


图 2.1-(18) 安装语言支持

安装完成后，点击语言支持选项卡中的“Install / Remove Languages...”：

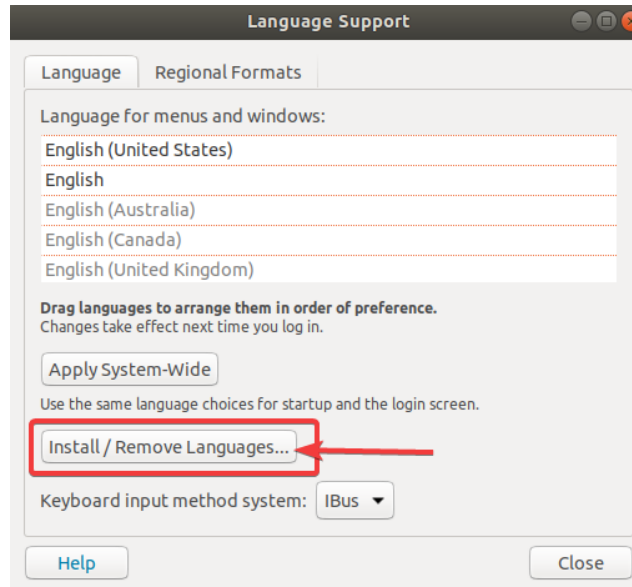


图 2.1-(19) 选择安装/移除语言

在菜单中找到“Chinese(simplified)”并在后方可选框中打勾，然后点击“Apply”：

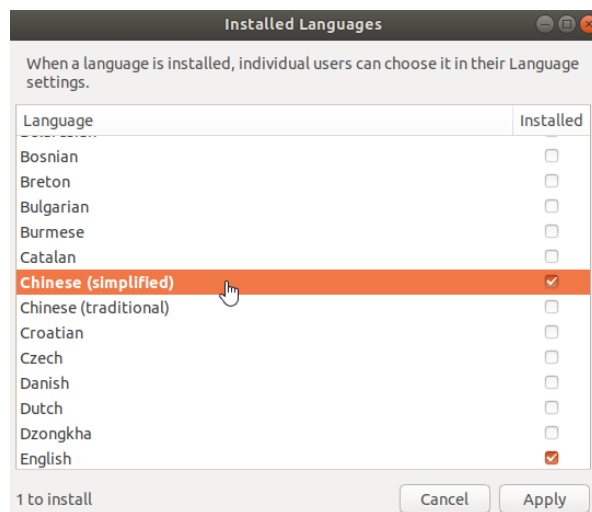


图 2.1-(20) 选择中文语言

等待安装完成后重新启动系统，然后重新进入图 2.1-(17)界面，点击右侧“Language”，然后选择汉语，勾选后重启应用即可：

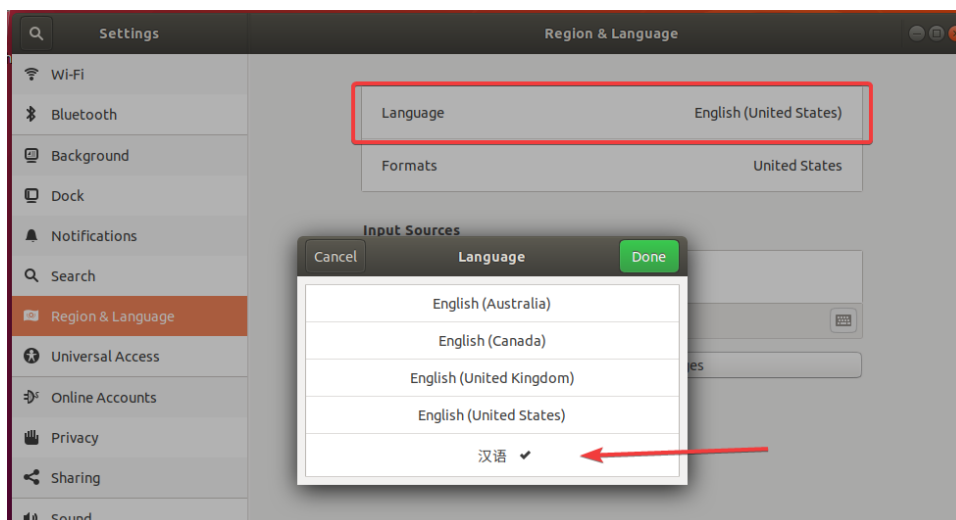


图 2.1-(21) 应用中文

至此就完成了虚拟机的安装。接下来我们还需要配置编译环境、推流环境以及云计算盒的连接与配置。

2.2 编译环境搭建

在 Ubuntu 开发环境下更新 apt 安装工具的软件包目录、安装文件解压工具，以便于后续利用 apt 安装程序与解压压缩包。

首先右键单击桌面，选择“进入终端”，输入以下命令（“#”号及其后面内容为注释，不输入）并按回车结束输入：

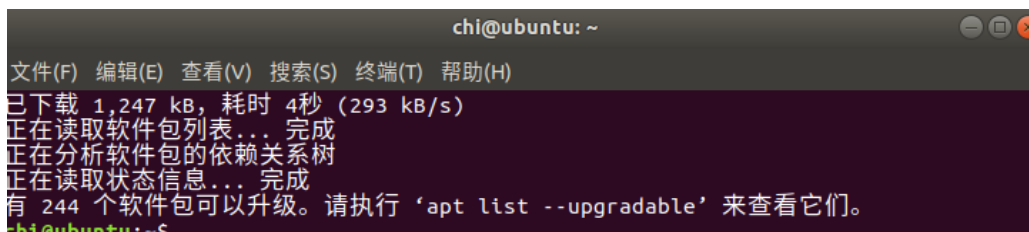
```
sudo apt update # 更新 apt 安装工具的软件包目录
```

由于这里并不是以管理员用户登录，所以此时系统要求输入上一步中配置虚拟环境时输入的用户密码。输入时没有提示，“回车”键结束输入：

```
chi@ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
chi@ubuntu:~$ sudo apt update  
[sudo] chi 的密码:
```

图 2.2-(1) 终端输入 apt 更新命令

等待下载完成，提示以下信息则表明更新完成：

A terminal window titled 'chi@ubuntu: ~' showing the output of an 'apt update' command. The output indicates that 1,247 kB were downloaded in 4 seconds at a rate of 293 kB/s. It also shows that the software package lists were read, the dependency tree was analyzed, and status information was read, all successfully. Finally, it states that 244 software packages can be upgraded and suggests running 'apt list --upgradable' to view them.

```
chi@ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
已下载 1,247 kB, 耗时 4秒 (293 kB/s)  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
正在读取状态信息... 完成  
有 244 个软件包可以升级。请执行 'apt list --upgradable' 来查看它们。  
chi@ubuntu:~$
```

图 2.2-(2) apt 更新完成

接下来安装解压工具，在终端中继续输入：

```
sudo apt install unzip # 安装解压工具
```

Ubuntu 系统内置了 unzip 解压工具，所以此命令只是对 unzip 进行更新。提示以下信息则表明更新完成：

A terminal window titled 'chi@ubuntu: ~' showing the output of 'sudo apt install unzip'. It lists the software packages to be upgraded (unzip) and shows that 1 package was upgraded, 0 were newly installed, 0 were to be removed, and 243 were not upgraded. It details the download of a 168 kB archive, the space it will consume, and the source URL. It then shows the download progress (168 kB, 1 second, 336 kB/s) and the process of reading the database, preparing to unpack the file, and finally setting up the triggers for mime-support and man-db.

```
chi@ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
下列软件包将被升级：  
unzip  
升级了 1 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 243 个软件包未被升级。  
需要下载 168 kB 的归档。  
解压缩后会消耗 0 B 的额外空间。  
获取:1 http://mirrors.aliyun.com/ubuntu bionic-updates/main amd64 unzip amd64 6.0-21ubuntu1.2 [168 kB]  
已下载 168 kB, 耗时 1秒 (336 kB/s)  
(正在读取数据库 ... 系统当前共安装有 116402 个文件和目录。)  
正准备解包 .../unzip_6.0-21ubuntu1.2_amd64.deb ...  
正在将 unzip (6.0-21ubuntu1.2) 解包到 (6.0-21ubuntu1.1) 上 ...  
正在设置 unzip (6.0-21ubuntu1.2) ...  
正在处理用于 mime-support (3.60ubuntu1) 的触发器 ...  
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...  
chi@ubuntu:~$
```

图 2.2-(3) 解压工具更新完成

在后文中所提到的输入命令均为在终端内输入命令。

接下来就可以下载编译环境的依赖文件了。首先进入算能官网，选择服务与支持中的技术资料，在页面的选项卡中选择资料下载，在左侧列表选择 SDK 中的 V3.0.0 下载 SOPHONSDK 3.0.0(https://sophon-file.sophon.cn/sophon-prod-s3/drive/22/07/18/11/sophonsdk_v3.0.0_20220716.zip)。

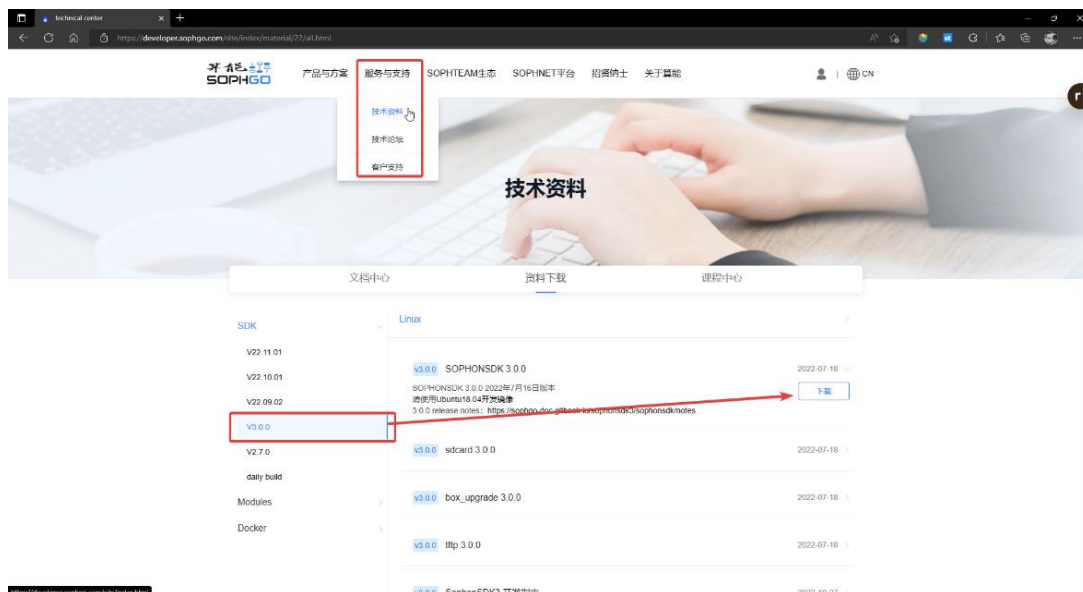


图 2.2-(4) 算能 SDK 下载

VMware workstation 在安装 Ubuntu 系统时内置了 VMware Tools，此插件的功能之一是支持将文件从本地系统中拖动到虚拟机中完成复制操作。所以可以在本地环境先下载 SDK，然后复制到虚拟环境中，这样在之后整理依赖包时可以在本地系统中更直观地进行管理。若想直接在虚拟机中管理，则可以直接在 Ubuntu 系统内置的 Firefox 浏览器中下载。

然后将 SDK 解压，将里面的文件移动到本地编译系统下的用户主目录。注意，刚下载的压缩包（后缀名为.zip）内还有一个压缩包（后缀名为.tar.gz）这里需要用到两条不同的命令。

在压缩包所在文件夹下单击右键进入终端，解压第一个压缩包（后缀名为.zip）的命令如下：

```
# unzip <sdk_zip_file>.zip
# 上面下载的:sophonsdk_v3.0.0_20220716.zip
unzip sophonsdk_v3.@.0_20220716.zip #解压 zip 包
```

解压完成效果如下：

```
chi@ubuntu:~$ unzip sophonsdk_v3.0.0_20220716.zip
Archive:  sophonsdk_v3.0.0_20220716.zip
  creating:  sophonsdk_v3.0.0_20220716/
 inflating:  sophonsdk_v3.0.0_20220716/sophonsdk.MD5
 inflating:  __MACOSX/sophonsdk_v3.0.0_20220716/._sophonsdk.MD5
 inflating:  sophonsdk_v3.0.0_20220716/release_version.txt
 inflating:  __MACOSX/sophonsdk_v3.0.0_20220716/._release_version.txt
 inflating:  sophonsdk_v3.0.0_20220716/sophonsdk_v3.0.0.tar.gz
 inflating:  __MACOSX/sophonsdk_v3.0.0_20220716/._sophonsdk_v3.0.0.tar.gz
```

图 2.2-(5) 算能 SDK 解压成功

进入解压后的目录，内有如下文件：

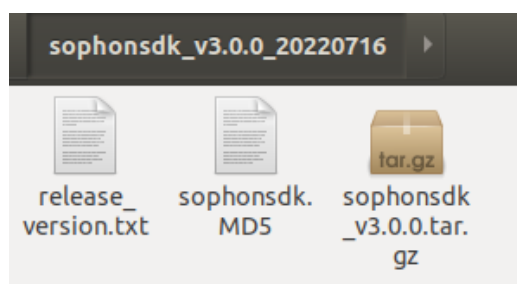


图 2.2-(6) 算能 SDK 解压后得到的文件

解压第二个压缩包（后缀名为.tar.gz）的命令如下：

```
# tar zxvf <sdk_files.tar.gz
# zip 包解压得到的文件为 sophonsdk_v3.0.0.tar.gz
# release 为 sdk 更新信息,MD5 为验证文件，这里用不到
tar zxvf sophonsdk_v3.0.0.tar.gz # 解压 tar.gz
```

将解压后的文件夹移动到用户主目录下，即完成了算能 SDK 的解压。

接下来下载 QTLib。访问 Sophon-qt 仓库(<https://github.com/sophon-ai-algo/sophon-qt/tree/main/qt-lib>)：

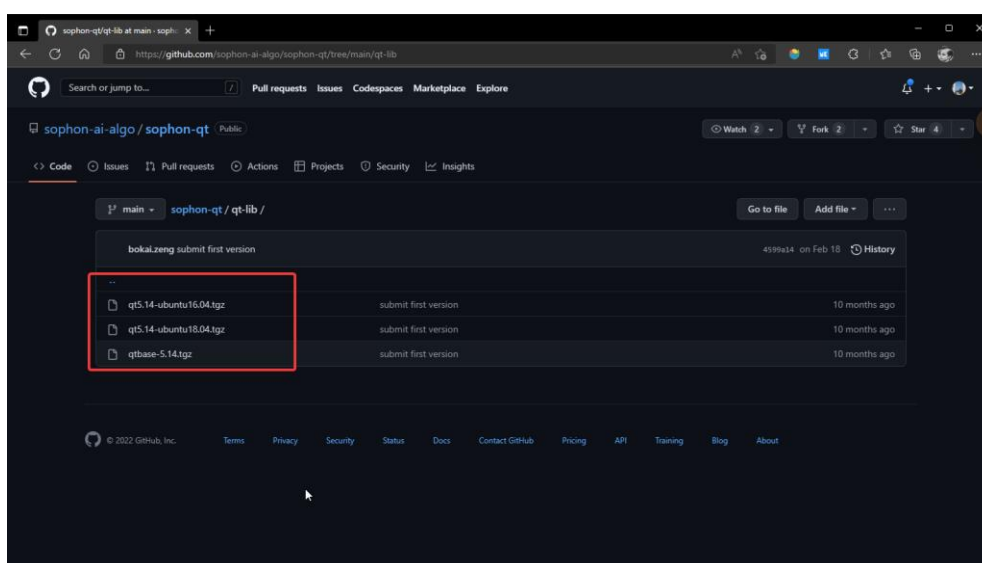


图 2.2-(7) QTLib 下载界面

根据当前编译系统的 ubuntu 版本不同 选择不同的 QTLib 压缩包(16.04 选择 qt5.14-ubuntu16.04.tgz ， 18.04 选择 qt5.14-ubuntu18.04.tgz ， 20.04 选择 qtbases-5.14.tgz)，下载并解压缩至用户主目录：

```
tar -zxvf qt5.14-ubuntu18.04.tgz # QTLib 的解压命令
```

解压后获得 `install` 文件。此文件为在 X86 环境下编译所需的 QTLib 文件。

为了使系统支持使用了 QTLib 库的程序的正常编译，还需要安装对应的依赖文件。

安装 QT 依赖 `qtbase5-dev`:

```
sudo apt install qtbase5-dev # 安装 QT 依赖 qtbase5-dev
```

由于 Tracker 功能需要安装 Eigen 依赖，安装 `libeigen3-dev`:

```
sudo apt-get install -y libeigen3-dev # 安装 libeigen3-dev
```

安装 `libgoogle-glog-dev` 与 `libexiv2-dev`:

```
sudo apt-get install -y libgoogle-glog-dev libexiv2-dev # 安装 glog/exiv2 依赖
```

我们需要在本地环境下编译出能够在 SE5 AI 计算盒中运行的程序，因此需要搭建交叉编译环境，在 Ubuntu18.04 的本地系统中安装 GCC 交叉编译器(<http://219.142.246.77:65000/sharing/49o1G1c5i>)。下载完成后，为保证编译器能在全局范围内生效，将编译器 `gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu` 解压至系统根目录下:

```
sudo tar -xvf gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz -C /  
# 代码尾部的"-C"参数的意思是自定义解压目录。“/”为 Ubuntu 系统根目录
```

解压完成后，还需配置环境变量，让交叉编译器全局有效。此处还需要配置刚才解压的算能 SDK 与编译器的环境变量。在 Ubuntu 下用户主目录下查看隐藏的 `.bashrc` 文件:

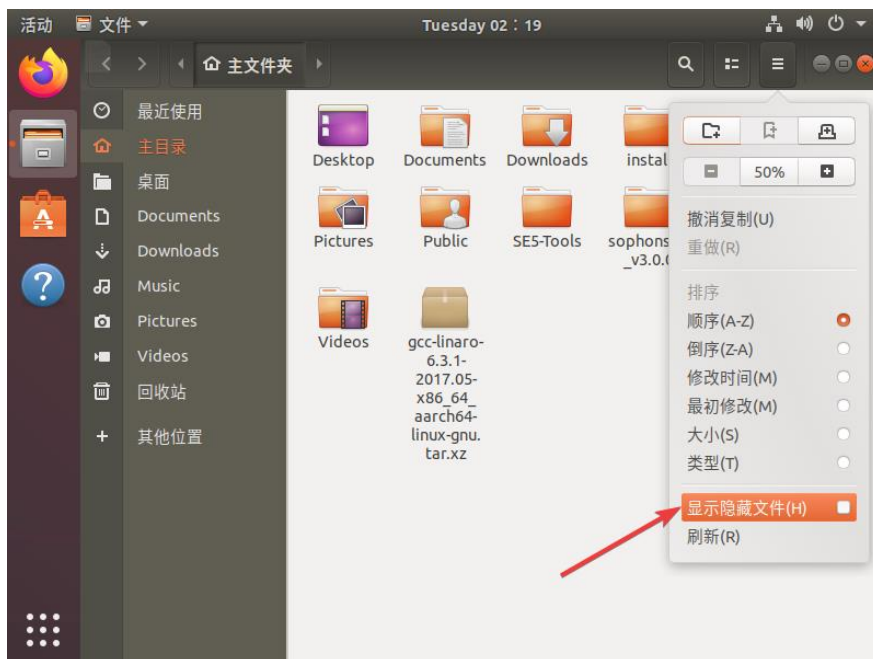


图 2.2-(8) 查看隐藏文件

可以在目录中看到.bashrc 文件。双击打开后在末尾添加如下代码：

```
#编译器路径(之前解压至根目录,所以此处不用修改)
export
PATH=/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++:/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc:$PATH
#算能 SDK 路径(此处根据实际位置将$sophonsdk_dir 修改为 SophonSDK 根路径。如
REL_TOP=/home/chi/sophonsdk_v3.0.0)
export REL_TOP=$sophonsdk_dir
```

添加完成后记得点击右上角的保存按钮。

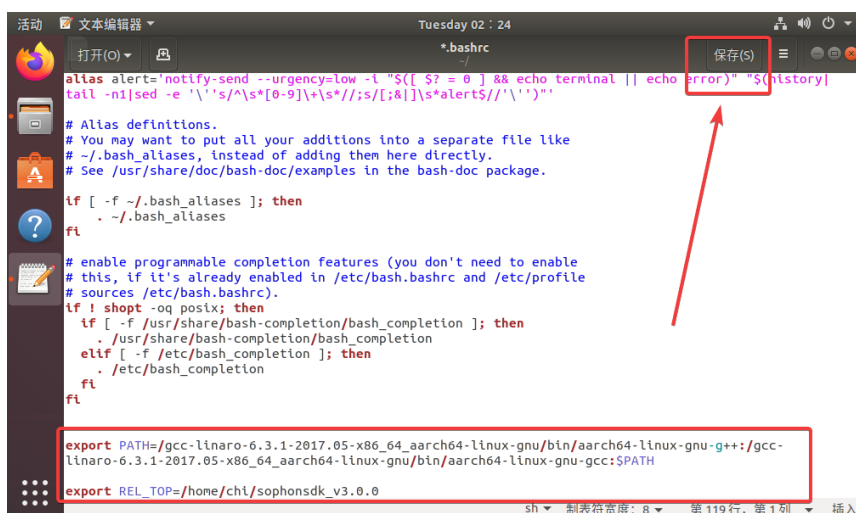


图 2.2-(9) 环境变量配置

至此，编译环境的搭建就结束了。若后续编译过程中出现错误，请首先排查编译环境是否配置完整且正确。

2.3 推流环境搭建

进入 EasyDarwin 的 GitHub 主页（<https://github.com/EasyDarwin/EasyDarwin/releases>），根据本地系统的类型进行下载，以下演示均以 Windows 为例。

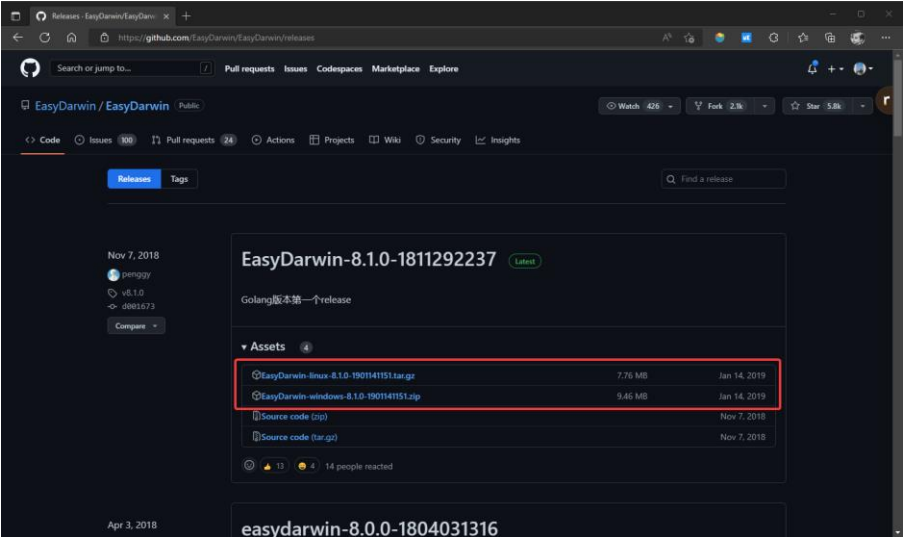


图 2.3-(1) EasyDarwin 的 GitHub 主页

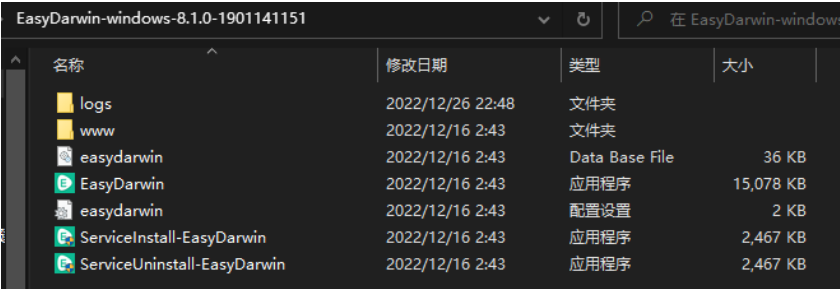


图 2.3-(2) EasyDarwin 下载得到的文件

此软件无需安装，只需在配置文件 `easydarwin.ini` 中进行简单的配置即可使用。主要的配置项如下：

配置项	说明
Port (http)	服务器后台端口号（默认为 10008）
Default_username	用户名（默认为 admin）
Default_password	密码（默认为 admin）
Port(rtsp)	rtsp 推流端口号（默认为 554）

表 2.3-(1) EasyDarwin 配置

如无特殊需求，则不建议修改配置文件。配置完成后，双击 `EasyDarwin.exe` 启动服务器，看到如下画面：



图 2.3-(3) EasyDarwin 运行界面

在浏览器中打开 <http://127.0.0.1:10008> 进入后台（此处若在配置文件中修改了 http 端口号，需自行修改链接中的端口号），并输入配置文件中的用户名和密码（没修改过配置文件则使用上表中的默认用户名与密码）若能成功登入后台（见下图），则推流服务器搭建完成。

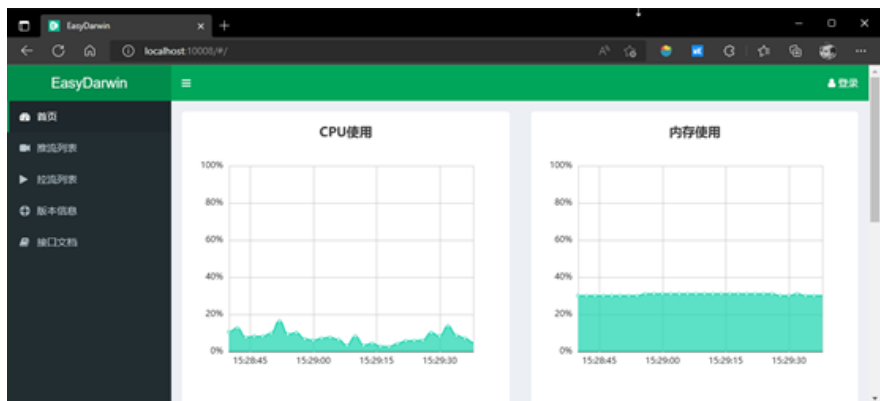


图 2.3-(4) EasyDarwin 后台界面

接下来进入 FFMPEG 官网（<http://ffmpeg.org/download.html>）下载 FFMPEG 作为推流工具：

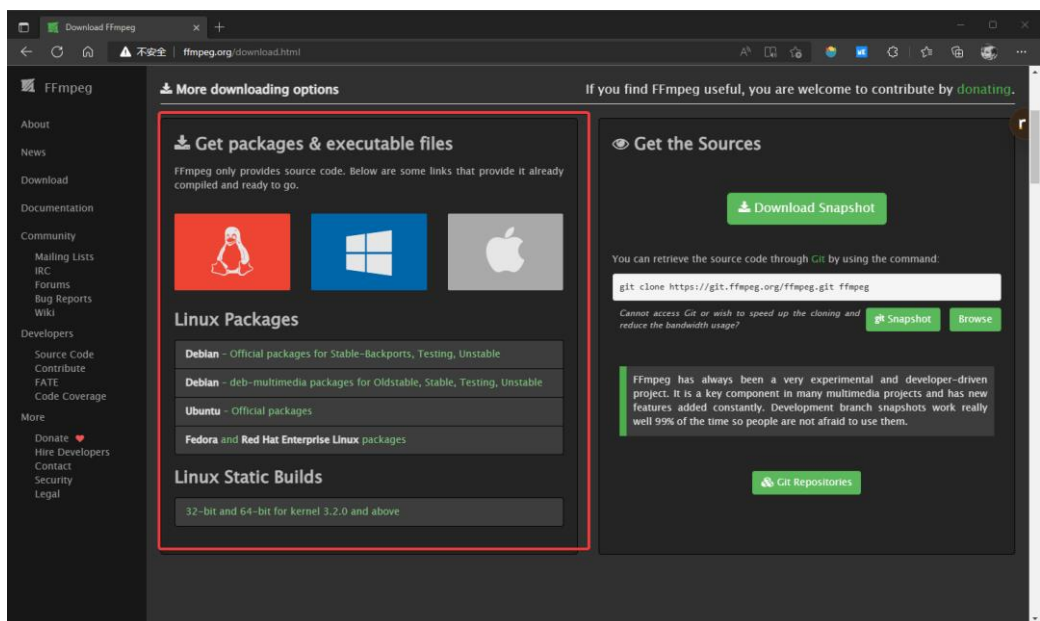


图 2.3-(5) FFMPEG 官网界面

下载完成后解压得到如下文件：

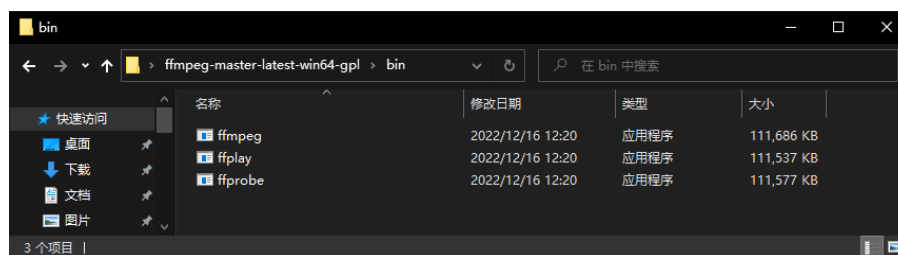


图 2.3-(6) FFMPEG 解压后的可执行文件

这里我们仅使用 **ffmpeg** 文件，使用时需要切换至 **ffmpeg** 下的 **bin** 目录，并在命令行内执行推流命令以调用。下面对 **ffmpeg** 的推流命令进行解释：

```
# ffmpeg -re -i <视频地址> -rtsp_transport <传输方式 tcp/udp> -vcodec h264 -f rtsp rtsp://<服务器 IP>/<子路径>#-re 表示按照帧率发送,否则 ffmpeg 会按照最高速率向流媒体服务器发送数据
# -i filename 指定输入文件名
# -vcodec codec 强制使用 codec 编解码《若填写"copy"代表不进行重新编码》:-f fmt 指定视频或音频的格式
# 本实验用例:
ffmpeg -re -i c:\Users\RDG\Desktop\out.mp4 -rtsp_transport tcp -vcodec h264 -f
rtsprtsp://127.0.0.1:554/test
#即将 C:\Users\ROG\Desktop\out.mp4 文件推流，播放地址为 rtsp://127.0.0.1:554/test
```

按住“win 键”+“R”，打开“运行窗口”，输入“cmd”，点击确定进入命令提示符：

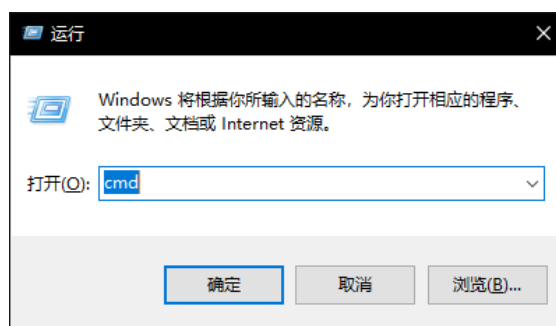


图 2.3-(7) “运行”窗口

然后使用“**cd <路径>**”进入解压的 FFMPEG 文件所在路径

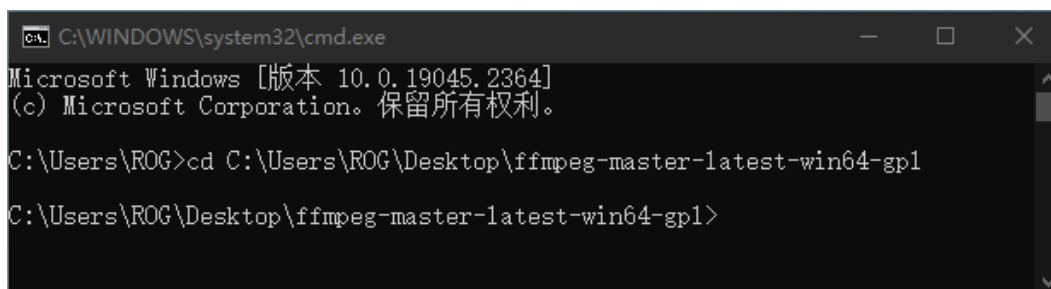


图 2.3-(8) cd 命令演示

输入上述代码中的推流命令后回车执行，得到以下运行效果：

```
CA\WINDOWS\system32\cmd.exe - ffmpeg -re -i C:\Users\ROG\Desktop\out.mp4 -rtsp_transport tcp -vcodec h264 -f rtsp rtsp://127.0.0.1:554/test/
14953 kb/s, 29.97 fps, 29.97 tbr, 30k tbn (default)
Metadata:
  handler_name      : Ambarella AVC
  vendor_id         : [0][0][0][0]
  encoder           : Ambarella AVC encoder
Stream #0:1[0x2](eng): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 128 kb/s (default)
Metadata:
  handler_name      : Ambarella AAC
  vendor_id         : [0][0][0][0]
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:1 -> #0:1 (aac (native) -> aac (native))
Press [q] to stop, [?] for help
[libx264 @ 000001b59deef4c0] using SAR=1/1
[libx264 @ 000001b59deef4c0] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
[libx264 @ 000001b59deef4c0] profile High, level 4.0, 4:2:0, 8-bit
[libx264 @ 000001b59deef4c0] 264 - core 164 - H.264/MPEG-4 AVC codec - Copyleft 2003-2022 - http://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=18 lookahead_threads=3 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=250 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0 qpmay=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, rtsp, to 'rtsp://127.0.0.1:554/test':
Metadata:
  major_brand       : isom
  minor_version     : 512
  compatible_brands: isomiso2avc1mp41
  encoder           : Lavf59.34.102
Stream #0:0(eng): Video: h264, yuvj420p(pc, progressive), 1920x1080 [SAR 1:1 DAR 16:9], q=2-31, 29.97 fps, 90k tbn (default)
Metadata:
  handler_name      : Ambarella AVC
  vendor_id         : [0][0][0][0]
  encoder           : Lavc59.55.100 libx264
Side data:
  cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: N/A
Stream #0:1(eng): Audio: aac (LC), 48000 Hz, stereo, fltp, 128 kb/s (default)
Metadata:
  handler_name      : Ambarella AAC
  vendor_id         : [0][0][0][0]
  encoder           : Lavc59.55.100 aac
frame= 491 fps= 25 q=29.0 size=N/A time=00:00:18.79 bitrate=N/A speed=0.957x
```

图 2.3-(9)启动 FFMPEG 并进行推流

此时可以看到推流已成功开始，登录 EasyDarwin 的后台界面查看是否成功推流至 EasyDarwin 服务器：

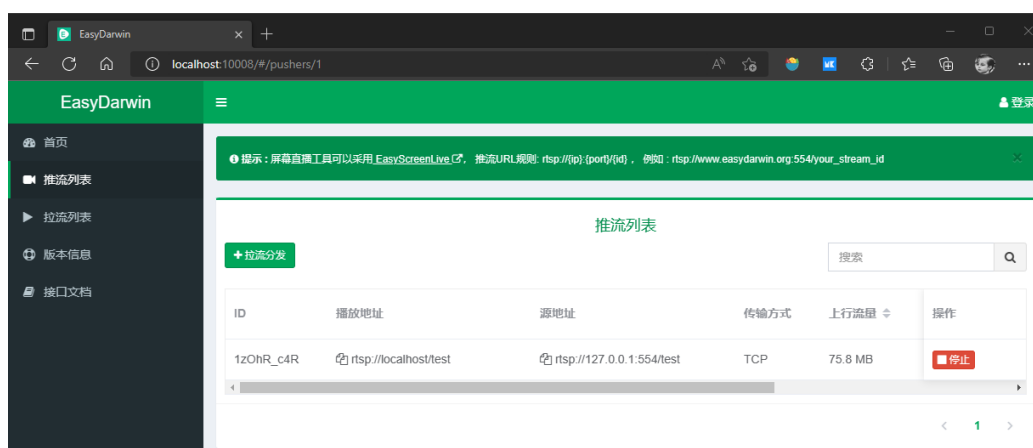


图 2.3-(10) 开始推流后 EasyDarwin 后台推流列表

我们可以使用 VLC media player 软件测试是否推流成功。先在 VLC 官网（<https://www.videolan.org/>）下载软件，安装完成后启动，在选项卡中选择媒体->打开网络串流：

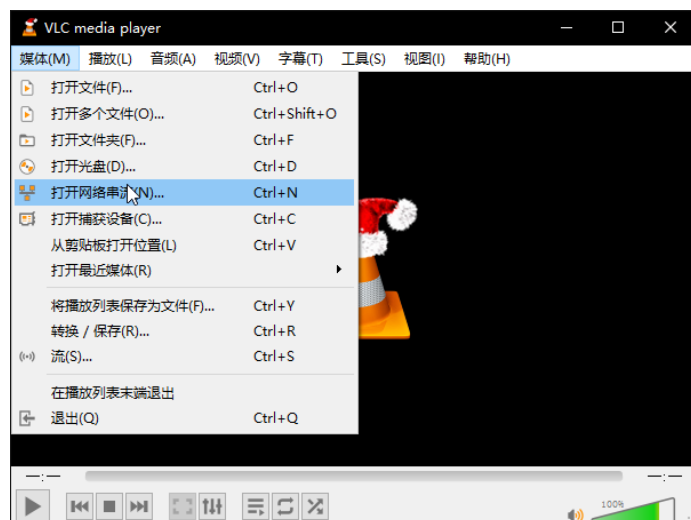


图 2.3-(11) VLC 选项卡选择

在网络 URL 中填入刚才在 ffmpeg 执行命令中的播放地址：



图 2.3-(12) VLC 推流地址填写

点击播放后就可以看到所推流的实时画面了：



图 2.3-(13) VLC 推流成功画面

此处需要注意，当推流视频播放完毕后，推流将会自动结束。若需继续推流，需要重新在推流机器中的命令行内重新执行上述的 ffmpeg 命令。

2.4 SE5 计算盒的连接

本地环境配置完成后，还需将 SE5 盒子接入局域网络。硬件方面，只需用网线将 SE5 盒子后方的 LAN 口与路由器的 LAN 口相连即可。



图 2.4-(1) SE5 盒子与路由器接线示意图

连接完成后，在浏览器中输入 SE5 计算盒的后台登陆地址（默认为 192.168.150.1），若能成功进入如下界面，则表明 SE5 计算盒已成功连入局域网。

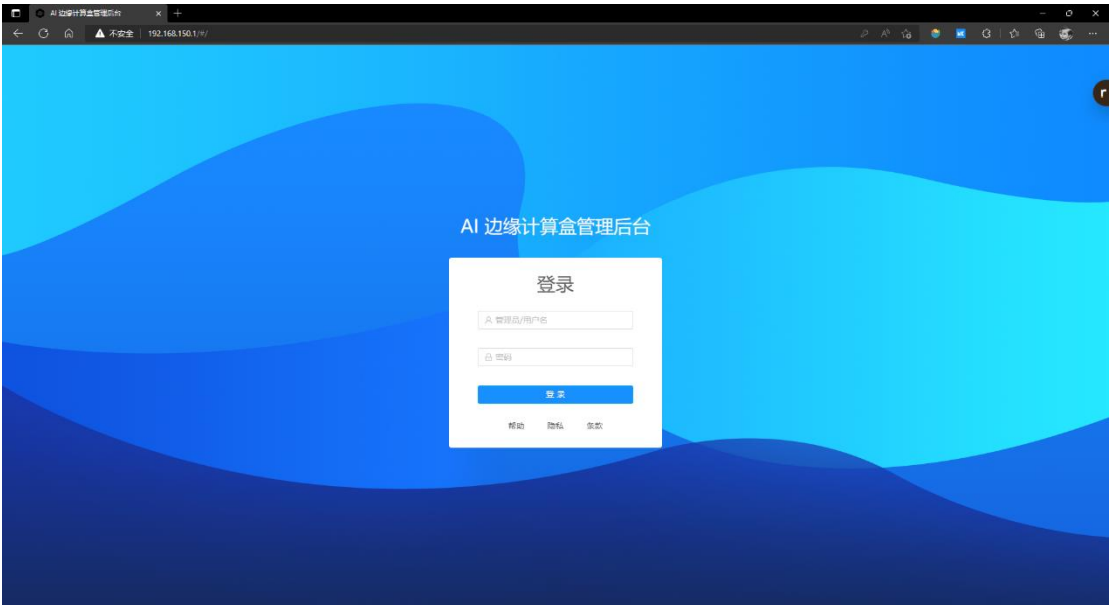


图 2.4-(2) SE5 盒子后台界面

若浏览器提示连接不可用，则表明路由器没有成功给计算盒分配正确的 IP 地址。此时需要进入路由器的后台界面调整 LAN 口的 IP 地址分配范围为 192.168.150.1 至 192.168.150.254。还需注意的是，不能和分配给现有连接设备的 IP 地址冲突。必须保证计算盒的登陆地址所用的 IP 不能被占用。路由器具体的设置方式因路由器的不同品牌及型号而异，请参照所用路由器的说明书设置。

接下来我们选择使用 FileZilla 软件作为 SE5 计算盒的文件管理软件，这有助于后期对计算盒中的文件能够有更直观的管理。

进入 FileZilla 官网（<https://filezilla-project.org/index.php>）下载最新安装包（client 版本）并进行安装，然后打开软件，看到如下界面：

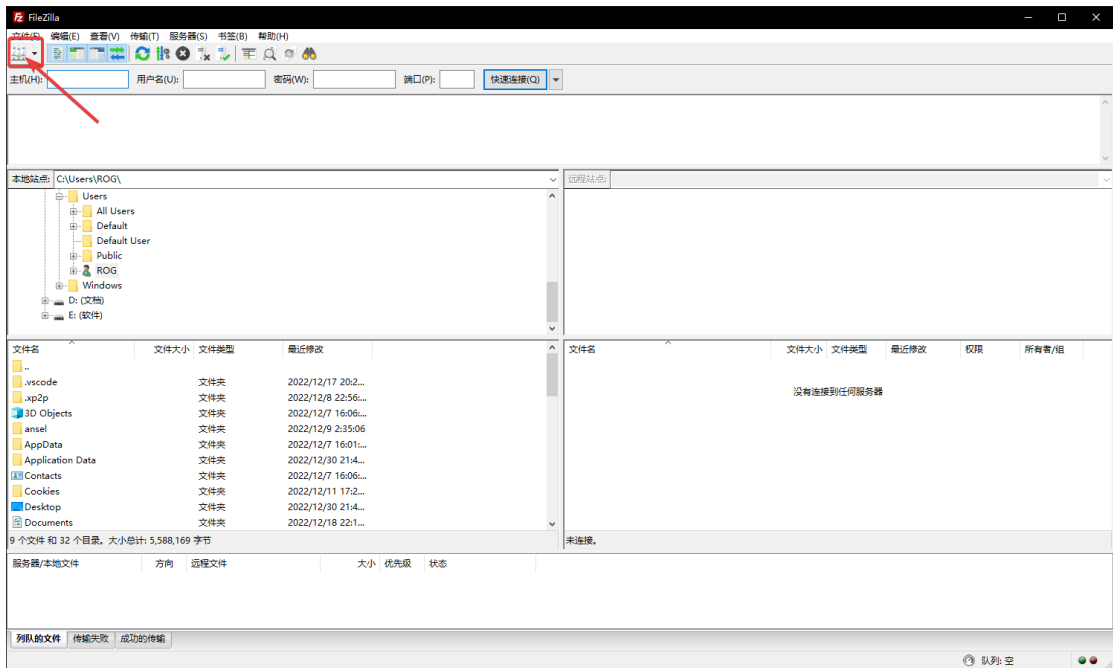


图 2.4- (3) filezilla 主界面

点击如上图箭头所指的图标，以添加 SE5 计算盒，弹出站点管理器，按照下图进行填写（密码同为 admin）：



图 2.4- (4) filezilla 站点管理器

填写完成后点击下方“连接”选项，上方提示框显示“列出 [用户目录] 的目录成功”即为连接成功：



图 2.4-(5) filezilla 连接成功

接下来需要在 SE5 计算盒中执行命令，选择使用 Xshell 这款软件来作为命令行远程终端软件，这样省去了每次执行命令都需要进入后台界面，在网页端进行操作的麻烦。

首先进入 Xshell 官方网站(<https://www.xshell.com/zh/free-for-home-school/>)，填入个人姓名和邮件地址，然后勾选“只需 Xshell”。然后登录到所填写的邮箱中，即可看到 Xshell 官方发送的邮件，内含 Xshell 的家庭版下载链接，点击下载并安装即可。

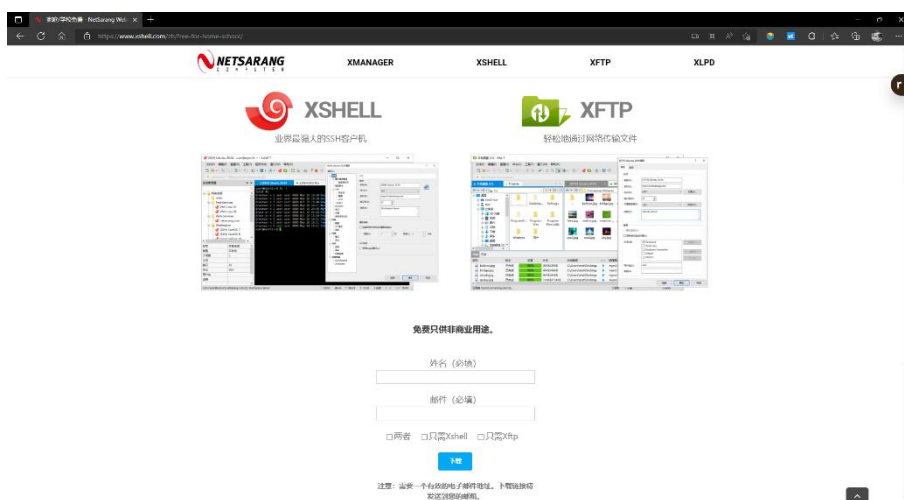


图 2.4-(5) Xshell 官网下载界面

进入软件后点击左上角添加主机，然后按照下图所示进行填写，完成后点击连接即可按照提示输入 SE5 盒子的用户名及密码即可（默认均为 admin）。

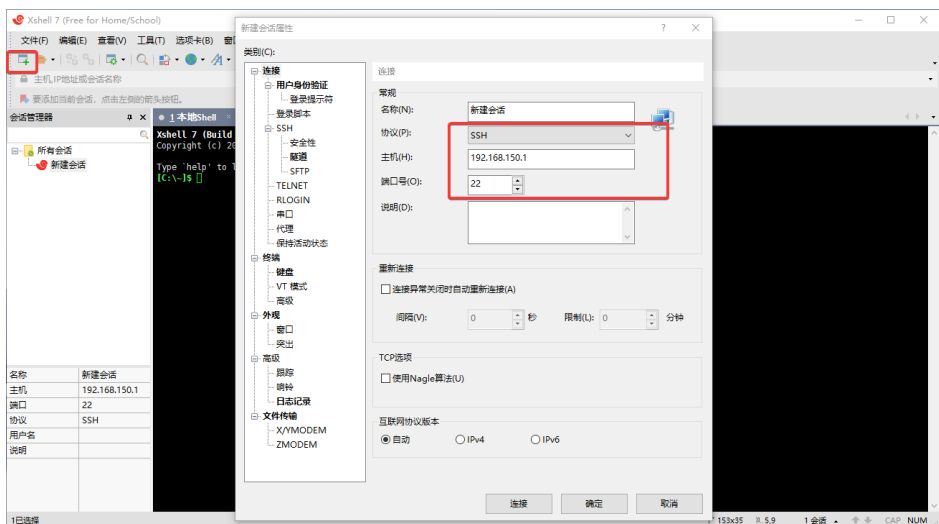


图 2.4-(6) Xshell 配置界面

至此基础环境配置就完成了，接下来进入到程序的编译及运行环节。

3. 程序的编译及运行

3.1 程序的编译

上述环境配置完成后就可以开始程序的编译了。将程序源代码解压至用户主目录，得到以下文件：

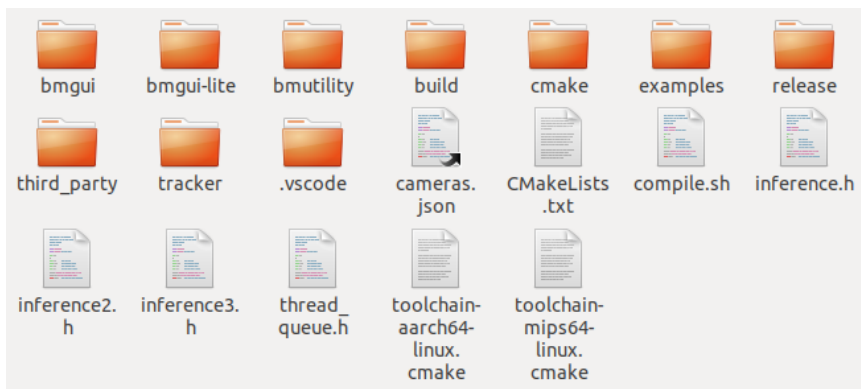


图 3.1-(1) 源代码目录

然后打开终端，为源代码文件夹赋予权限。先使用“cd”命令定位到源代码文件夹的上一层文件夹，然后输入：

```
sudo chmod -R 777 visualPerception/ # 赋予当前用户对目标文件夹及其内部文件可读可写权限
```

回车执行后，输入管理员密码。接下来若没有任何提示，则表明命令执行成功。

注意，此步骤非常重要，绝大多数出现错误的原因均是由于没有成功赋予对应文件夹权限。在后续上传文件至 SE5 盒子时还需要配置权限。

首先配置 QtLib 的路径（在上一节中解压得到的 install 文件夹路径）。打开

CMakeLists.txt 文件，找到以下代码段，并按照注释进行替换和修改：

```
if (USE_QTGUI)
if (${TARGET_ARCH} STREQUAL "soc")
set (QTDIR /opt/qt5.14-18.04-soc) # 将此路径更改为前一步骤 qt-lib 解压后根目录
set (Qt5widgets_DIR /opt/qt5.14-18.04-soc/lib/cmake/Qt5widgets)
# 将此路径更改为 qt-lib 解压后根目录下的 lib/cmake/Qt5widgets 文件夹
```

由于需要在 SE5 的 HDMI 输出最终识别结果，所以需要修改负责 HDMI 显示的参数。若不需要 HDMI 显示，则可跳过该步骤。

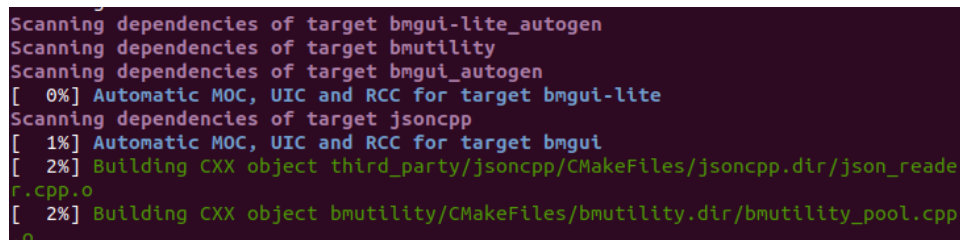
打开 compile.sh 文件，跳转到第 34 行。可以看到此行末尾有一个“-DUSEQTGUI=OFF”选项。此参数就可控制是否通过 HDMI 输出识别结果。脚本中此参数的默认值为 OFF，将其更改为 ON 即可。

```
cmake_params="-DTARGET_ARCH=target_arch -DUSEQTGUI=ON" # 设置参数，将 OFF 改为 ON
```

然后在终端内进入程序源代码所在目录，输入编译命令：

```
bash ./compile.sh soc # 编译指令
```

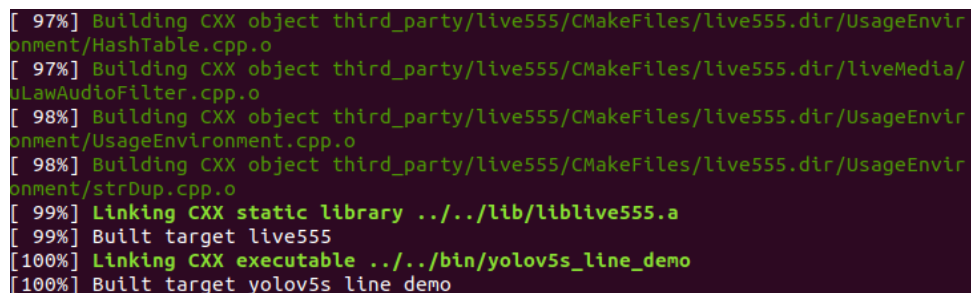
终端出现如下提示时即为正在编译。此过程中请勿执行其它操作。若在此过程中出现错误，请首先检查是否给源代码目录赋予权限：



```
Scanning dependencies of target bmgui-lite_autogen
Scanning dependencies of target bmutility
Scanning dependencies of target bmgui_autogen
[ 0%] Automatic MOC, UIC and RCC for target bmgui-lite
Scanning dependencies of target jsoncpp
[ 1%] Automatic MOC, UIC and RCC for target bmgui
[ 2%] Building CXX object third_party/jsoncpp/CMakeFiles/jsoncpp.dir/json_rea
r.cpp.o
[ 2%] Building CXX object bmutility/CMakeFiles/bmutility.dir/bmutility_pool.cpp
.o
```

图 3.1-(2) 编译过程

进度条到达 100%后编译即完成：



```
[ 97%] Building CXX object third_party/live555/CMakeFiles/live555.dir/UsageEnvir
onment/HashTable.cpp.o
[ 97%] Building CXX object third_party/live555/CMakeFiles/live555.dir/liveMedia/
LiveAudioFilter.cpp.o
[ 98%] Building CXX object third_party/live555/CMakeFiles/live555.dir/UsageEnvir
onment/UsageEnvironment.cpp.o
[ 98%] Building CXX object third_party/live555/CMakeFiles/live555.dir/UsageEnvir
onment/strDup.cpp.o
[ 99%] Linking CXX static library ../../lib/liblive555.a
[ 99%] Built target live555
[100%] Linking CXX executable ../../bin/yolov5s_line_demo
[100%] Built target yolov5s_line_demo
```

图 3.1-(3) 编译结束

编译完成后，编译好的文件将会生成到源代码目录下的 cmake-build-debug 目录内，可执行文件则存放在此目录下的 bin 文件夹内。

将修改过后的 QTLib 库（存放在项目文件 visual_perception_proj\source 下

install.zip，解压后文件夹名也为 install，注意不要与编译时使用的 QTLib 库搞混）与 cmake-build-debug 目录一同上传到 SE5 计算盒内。

打开 FileZilla 软件，在右侧选择路径为 “/home/admin”，然后将需上传的文件夹拖动至 admin 文件夹上，等待上传完成即可：

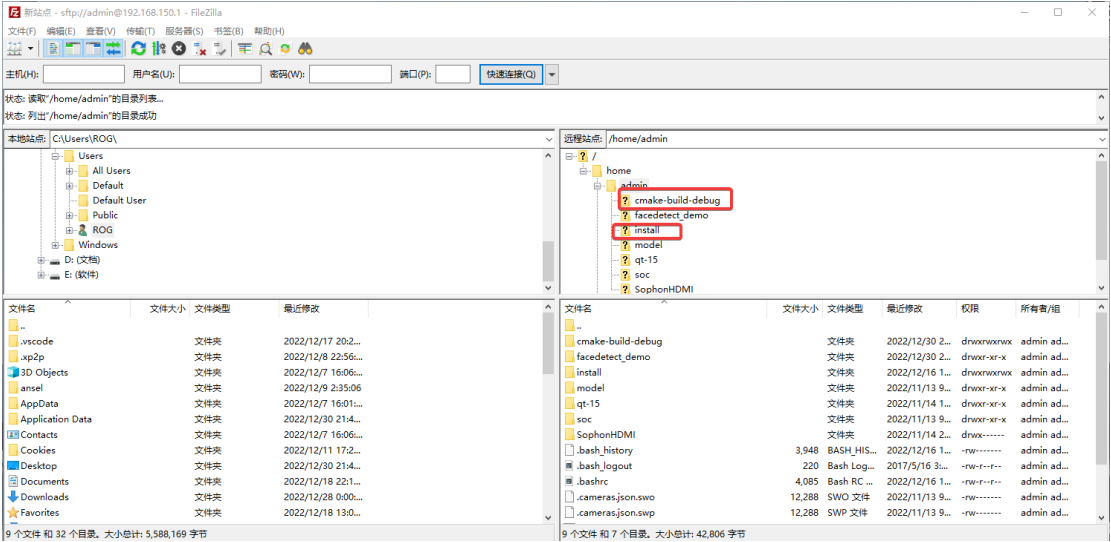


图 3.1-(4)文件上传

上传完成后，在本地新建脚本文件 run_hdmi_yolo_line.sh，文件内代码如下，注意按照注释进行路径的修改：

```
#!/bin/sh -x
fl2000=$(lsmod | grep fl2000 | awk '{print $1}')
echo $fl2000
if["$fl2000" != "fl2000" ]; then
echo "insmod fl2000"
else
echo "fl2000 already insmod "
fi
export PATH=$PATH:/system/bin:/bm_bin
export QTDIR=/home/admin/install/lib # qtlib 在系统上的路径 (根据需要进行修改)
export QT_QPA_FONTDIR=$QTDIR/fonts
export QT_QPA_PLATFORM_PLUGIN_PATH=/home/admin/install/plugins # plugins 在 install 下的路径
(根据需要进行修改)
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/admin/install/lib:/home/admin/cmake-build-debug/lib/
# 添加 qtlib 的路径与 bmlib 库的路径至环境变量 (以冒号分隔)
# bmlib 库路径为 cmake-build-debug 文件夹下的 lib 路径
export QT_QPA_PLATFORM=linuxfb:fb=/dev/f12000-0 # framebuffer 驱动
export QWS_MOUSE_PROTO=/dev/input/event3
chmod +x ./yolov5s_line_demo
./yolov5s_line_demo
```

编辑完成后，将其上传至 SE5 盒子中的“cmake-build-debug/bin”文件夹中，然后打开终端软件 Xshell，以 admin 的身份登录后进入上述路径，执行：

```
./yolov5s_line_demo -help
```

（若提示权限不足，则参照之前的赋权命令对 cmake-build-debug 文件夹授权。）

此命令用来查看命令行参数帮助信息，其中最重要的信息为“--bmodel”，即 bmodel 文件的存放路径，“--warningThresh”是距离报警阈值设定：

```
Usage: yolov5s_line_demo [params]

--bmodel (value:/data/models/yolov5s_1batch_fp32.bmodel)
    input bmodel path
--config (value:./cameras.json)
    path to cameras.json
--help (value:true)
    Print help information.
--max_batch (value:1)
    Max batch size
--num (value:1)
    Channels to run
--output (value:None)
    Output stream URL
--skip (value:1)
    skip N frames to detect
--warningThresh (value:10.0)
    warning threshold of distance
```

图 3.1-(5) 命令行帮助信息

然后在 FileZilla 中进入到此目录中（若没有就新建文件夹，若提示权限不足就在目标文件夹上单击右键，选择权限，然后将权限设置为 777，记得要选择“对其及其子目录与文件生效”选项），将先前准备好的深度学习模型上传至此。

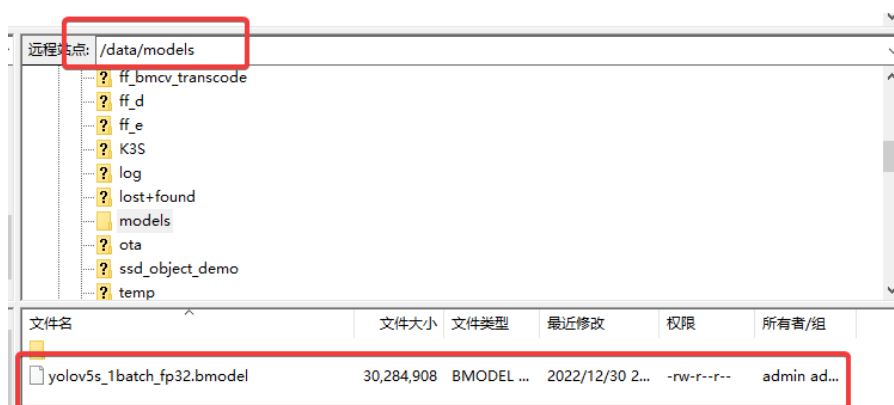


图 3.1-(6) 深度学习模型存放路径

完成上述步骤后，再新建 cameras.json 文件（address 修改为宿主机推流地址）：

```

{
  "max_show_windows": 16,
  "cards": [
    {
      "devid": 0,
      "cameras": [
        {
          "address": "rtsp://192.168.150.100:554/test",
          "chan_num": 2,
          "road_ROI": [
            [604,806],[912,536],[1033,528],[1396,804]],
          "HMat": [
            [1.6917548704398884,-1.5983741568609355,883.7178702266243],[-0.0131883242630214
05,-0.8237382886626196,756.7365746500346],[1.6910074589321605e-06,-0.0016404044446291432,
1.0]
          ]
        }
      ]
    }
  ],
  "pipeline": {
    "preprocess": {
      "thread_num": 4,
      "queue_size": 16
    },
    "inference": {
      "thread_num": 1,
      "queue_size": 16
    },
    "postprocess": {
      "thread_num": 4,
      "queue_size": 16
    }
  }
}

```

完成编辑并保存后也将其上传至“cmake-build-debug/bin”文件夹中。



图 3.1-(7) bin 文件夹内文件

3.2 程序的运行

完成上述步骤后，我们就可以在 SE5 盒子上运行程序了。

将 SE5 计算盒的 HDMI 接口与显示器相连接后，在 Xshell 中执行脚本：

```
sudo ./run_hdmi_show.sh # 程序运行
```

若执行后报错找不到库文件，可检查脚本 run_hdmi_yolo_line.sh 中的库文件路径是否为绝对路径。执行成功后终端出现以下信息：

```
Open stream rtsp://192.168.150.100:554/test
[rtsp @ 0x8c3660] method DESCRIBE failed: 404 NOT FOUND
Can't open file rtsp://192.168.150.100:554/test
no screens available, assuming 24-bit color
Cannot create window: no screens available
```

图 3.2-(1) 未启动推流的提示

此信息表明还未开始推流。依照 3.1.3 的推流环境配置，先启动 EasyDarwin 推流服务器，然后在本地启动命令行，使用 FFMPEG 进行推流。推流成功后显示如下：


```

Open stream rtsp://192.168.150.100:554/test
interleave TCP based rtp used: 0, 1
interleave TCP based rtp used: 2, 3
Init:total stream num:2
BMvidDecCreateW5 board id 0 coreid 0
libbmvideo.so addr : /system/lib/libbmvideo.so, name_len: 12
vpu firmware addr: /system/lib/vpu_firmware/chagall_dec.bin
VERSION=0, REVISION=213135
create video decoder ok!
decoder need more stream!
id=0, ffmpeg delayed frames: 0
decoder need more stream!
id=0, ffmpeg delayed frames: 1
decoder need more stream!
id=0, ffmpeg delayed frames: 2
id=0, ffmpeg delayed frames: 3
Open /dev/jpu successfully, device index = 0, jpu fd = 123, vpp fd = 124
[bmlib_runtime][info] driver version is 2.7.0

```

图 3.2-(2) 成功拉流后终端提示

此时就可以在显示器上看到经过 AI 计算后输出的画面了。



图 3.2-(3) 经过 AI 计算后的输出画面

可以看到，车道线以蓝色线条标识，车辆以矩形框进行标识，且在框上方标注了车辆的 id 与距离。当距离过近时，检测框标记为红色。

当推流视频结束后，FFMPEG 将会自动结束运行，但 SE5 AI 计算盒仍处于准备拉流状态：

```

[2022-12-31:22:29:13] total fps =0.0,ch=0: speed=0.0
[2022-12-31:22:29:14] total fps =0.0,ch=0: speed=0.0
[2022-12-31:22:29:15] total fps =0.0,ch=0: speed=0.0
[2022-12-31:22:29:16] total fps =0.0,ch=0: speed=0.0
[2022-12-31:22:29:17] total fps =0.0,ch=0: speed=0.0
[2022-12-31:22:29:18] total fps =0.0,ch=0: speed=0.0

```

图 3.2-(4) 等待拉流

此时可以在终端中使用快捷键“ctrl”+“c”来结束运行。