

飞机安装镜像+ipk+避障

by Zhang Xian

前言

飞机要实现避障功能，需要推入

- 镜像：有版本号
- ipk：有版本号
- Captain：有版本号
- AutoPilot：编译后的可执行文件，飞机位置 `/hover/PathPlanningNewo`
- libPathPlanning.so：编译后的可执行文件，飞机位置 `/hover/lib64/`
- CD_params：飞机里可直接修改，也可从repo里推入，飞机位置：`/hover/PathPlanningNewo`
- setting.yml：飞机里可直接修改，也可从repo里推入，飞机位置：`/hover/PathPlanningNeo`
- stereo.yaml：双目标定文件，双目标定一次后，推入，飞机位置：`/mnt/persist`

上述一一推入飞机后，需要

1. 用USB线连接PC和飞机，执行

```
adb shell
cd data/misc/wifi
vi hostapd.conf
```

修改飞机的 ID 和 飞机的连接密码，例如

```
ssid=hc2-496531           //hc2小写，否则小遥控器无法连接飞机，尾数自定义如 hc2-
b3-88也可以
wpa_passphrase=1234567890 //wifi密码
```

2. 手机打开 `Wifi` 设置，找到 `hc2-496531` 输入密码 `1234567890` 即可连接。若失败，换手机或重连；
3. 放飞分两种方式，手持或者从地面起飞。手持不行，就放地上起飞，具体可见 `APP` 或小遥控器的操作。

1 repo

本节操作全部在 PC 的 ubuntu 18.04 系统上完成。

1.1 依赖

- 【1】gcc 版本 7.0+，安装见[2]；
- 【2】cmake 3.0+，安装 建议方法2

#方法1：源码安装（不建议RC1, RC2,..., 因它们是release candidate, 即发行候选版本, 非正式版本）

```
wget "https://cmake.org/files/v3.14/cmake-3.14.5-Linux-x86_64.tar.gz"
```

```
tar xzf ./cmake-3.12.2-Linux-x86_64.tar.gz
```

```
sudo cp -rf ./cmake-3.12.2-Linux-x86_64/bin /usr/local/
```

```
sudo cp -rf ./cmake-3.12.2-Linux-x86_64/doc /usr/local/
```

```
sudo cp -rf ./cmake-3.12.2-Linux-x86_64/share /usr/local/
```

#方法2：ubuntu自带工具安装[3]

```
sudo apt-get -y install cmake
```

【3】安装 adb, 安装如下

```
wget https://dl.google.com/android/repository/platform-tools-latest-linux.zip
```

```
unzip ./platform-tools-latest-linux.zip
```

```
sudo cp platform-toos/adb /usr/bin/adb
```

```
sudo cp platform-tools/fastboot /usr/bin/fastboot
```

【4】安装 32 bit 的库支持（因编译工具链中有个别程序是 32-bit）

```
sudo dpkg --add-architecture i386
```

```
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
```

```
sudo apt-get install libtool
```

```
//sudo apt-get install -y ack-grep zsh vim curl unzip thefuck 取决于用户自己
```

【5】repo 库

```
wget http://git.zerozero.cn/zz-public/git-repo/raw/master/repo && chmod +x
```

```
repo && sudo mv repo /usr/bin
```

1.2 部署

【1】repo init

执行

```
mkdir repo && cd repo
```

```
repo init --no-repo-verify -u
```

```
git@git.zerozero.cn:HoverCamera2/product_repos.git -m hc2-master.xml -b  
feature-oademo
```

解释上述命令

//repo init：安装 repo 后需要初始化 repo, 否则在终端执行 repo version 会提示你安装 repo init

//--no-repo-verify：指定不检查 repo 仓库是否需要更新, 也可以放在命令末尾位置, 如 master 之后

//-u：指定 Manifest 文件的 Git 访问路径, 例如

```
git@git.zerozero.cn:HoverCamera2/product_repos.git
```

//-m：指定要使用的 Manifest 文件

//-b：指定要使用 Manifest 仓库中的某个特定分支

如果使用下面，

```
repo init --no-repo-verify -u
git@git.zerozero.cn:HoverCamera2/product_repos.git -m hc2-dev.xml -b master
```

会持续报错，原因并非是权限不够，而是过时了，因此建议用

```
repo init --no-repo-verify -u
git@git.zerozero.cn:HoverCamera2/product_repos.git -m hc2-master.xml -b
feature-oademo
```

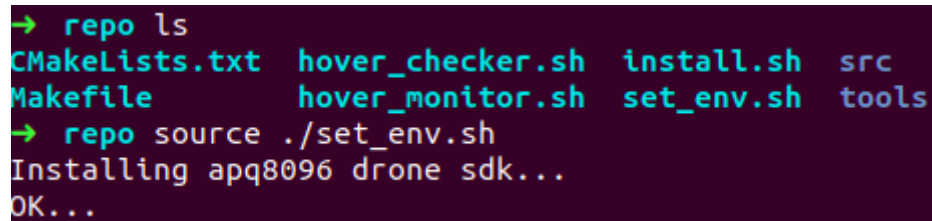
`feature-oademo` 不会报错。

【2】repo sync

如果报错，要咨询管理员打开权限，完了再sync，第一次时间较长，后面就好了。之后需要 `source` 一下，repo 目录下执行

```
source ./set_env.sh
```

成功后，界面如下



```
→ repo ls
CMakeLists.txt  hover_checker.sh  install.sh  src
Makefile        hover_monitor.sh  set_env.sh  tools
→ repo source ./set_env.sh
Installing apq8096 drone sdk...
OK...
```

这是公司嵌入式内部开发的代码，至此部署了编译环境。

1.3 编译代码

在 repo目录下

```
1 到避障代码路径
cd src/drone_platform/vision/Autopilot

2 根据需要修改代码，保存退出

3 cd 到 repo 目录下
cd build
ninja AutoPilot //AutoPilot包含整个cpp和头文件以及工程文件，其实来自 GitLab
release-hc2-pw分支
```

注意：`ninja AutoPilot` 相当于执行 `make AutoPilot`，整个过程没有 `cmake ..` 和 `make`，原因如1.2小节所说，公司内部开发环境，免去了 `cmake ..` 和 `make`。编译无误后，会在 `/src/drone_platform/vision/AutoPilot/cmake/aarch64` 目录下产生 `AutoPilot` 可执行文件，

```

→ drone_platform cd vision
→ vision ls
Autopilot  eis  rovio_hc2  vision

→ vision
→ vision cd Autopilot
→ Autopilot ls
CMakeFiles  cmake  cmake_install.cmake
→ Autopilot cd cmake
→ cmake ls
aarch64
→ cmake cd aarch64
→ aarch64 ls
AutoPilot          fc.grpc.pb.h          obstacle_avoidance.grpc.pb.cc
AutoPilot.debug    fc.pb.cc              obstacle_avoidance.grpc.pb.h
CMakeFiles         fc.pb.h              obstacle_avoidance.pb.cc
base.grpc.pb.cc    led.grpc.pb.cc        obstacle_avoidance.pb.h
base.grpc.pb.h     led.grpc.pb.h         vns.grpc.pb.cc
base.pb.cc         led.pb.cc             vns.grpc.pb.h
base.pb.h          led.pb.h              vns.pb.cc
cmake_install.cmake libPathPlanning.debug  vns.pb.h
fc.grpc.pb.cc     libPathPlanning.so

```

注意上面绿色的 `AutoPilot`，它就是要被推入飞机的文档。

1.4 推入飞机

用 USB线连接 PC，

```

adb shell
ls
cd hover
ls
cd PathPlanningNeo
ls

```

操作如下图

```

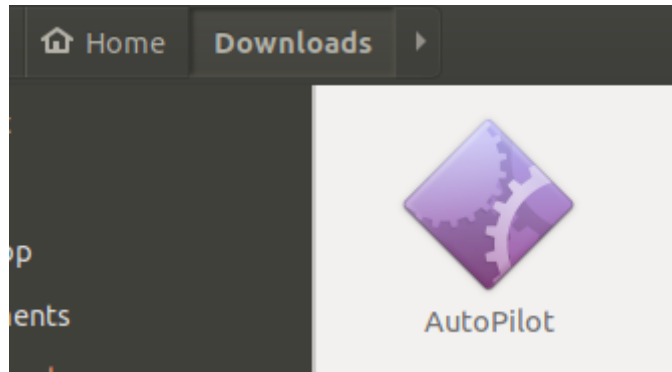
→ ~ adb shell
* daemon not running; starting now at tcp:5037
* daemon started successfully
/ # ls
DCIM          cache          firmware      linuxrc       proc          share          usr
WEBSERVER     data           home          lost+found    root          srv            var
bin           dev            hover         media         run           sys
boot         dsp            lib           mnt           sbin          system
build.prop    etc            lib64         persist       sdcard        tmp
/ # cd hover
~/application/1.1.95 # ls
Captain          camera_heater    gimbal_service.ipc  rc
PathPlanningNeo  camera_service  hover              rovio_hc2
PoseTracker      config          imu_heater         snpe_wrapper
battery_service  config.prototxt lib                stereo_cam
bin             confman         lib64              stereo_vision
bottom_cam       fc             libdsp            tests
bottom_vision    fpv            log               tools
calibration      front_vision    mediaserver        xbee_service
calibration_server gimbal_service  nanomsg-fastrpcd  zzlog_service
~/application/1.1.95 # cd PathPlanningNeo/
~/application/1.1.95/PathPlanningNeo # ls
AutoPilot        CD_params.yml    setting.yml
~/application/1.1.95/PathPlanningNeo # 

```

将步骤 1.3 中的绿色文件推入到 `/hover/PathPlanningNeo/` 即可，如果飞机里面有，应该会自动覆盖。保险起见，可以将其通过 `adb pull` 到电脑某个地方。例如重开个终端

```
adb pull /hover/PathPlanningNewo/AutoPilot /home/xian/Downloads
```

得到



小心路径，ubuntu默认 `/home/xian/Downloads` 中的 `/xian` 不会显示，建议到 `/Downloads` 目录下用 `pwd` 命令看看保险。

【怎么推入】

下面的是错误的：

```
adb push
/home/xian/workspace/repo/build/src/drone_platform/vision/Autopilot/cmake/aarc
h64 /hover/PathPlanningNeo/
```

正确的方法是：先停止AutoPilot服务，然后推入，注意路径。

```
adb shell systemctl stop zz_pathplanning
```

```
adb push
/home/xian/workspace/repo/build/src/drone_platform/vision/AutoPilot/cmake/aarc
h64/AutoPilot /hover/PathPlanningNeo/
```

```
adb push
/home/xian/workspace/repo/build/src/drone_platform/vision/Autopilot/cmake/aarc
h64/libPathPlanning.so /hover/lib64/
```

```
adb push
/home/xian/workspace/repo/build/src/drone_platform/vision/AutoPilot/CD_params.
yaml /hover/PathPlanningNeo/
```

```
adb reboot
```

2 双目标定

飞机双目标定所需软硬件环境

1. 标定板
2. 带有 Matlab2016 环境的 win10 电脑

3. repo

这里准备了个压缩工具包，请分别解压



2.1 标定用的镜像

不同于飞机程序里的镜像，这个镜像仅仅是用来标定的，必须刷 `raw8_vga_fastboot` 镜像，

参考网址：<https://zerozero.yuque.com/software-gwtj3/gucf4i/oan3y4>

在解压后的 `Raw8_VGA飞机双目标定教程` 里执行

```
cd /home/xian/Desktop/避障资料/Raw8飞机双目标定教程/Raw8_VGA飞机双目标定教程/raw8_vga_fastboot/out/fastboot_build/emmc  
  
./fastboot-all.sh
```

安装后重启即可。

2.2 标定用的ipk

将 `stereo_calibration` 解压后得到下图



终端到此路径，飞机和PC相连，执行

```
./install.sh
```

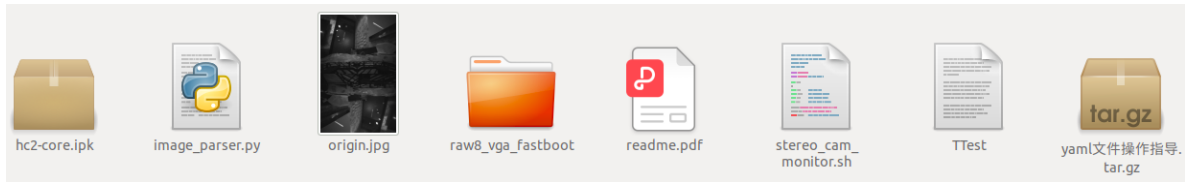
出现下面就OK

```
gimbal {  
  brake_sensitivity: 50  
  pitch_speed_sensitivity: 30  
}  
  
Save configuration to /persist/config.prototxt succeed.  
adb: error: cannot stat 'hover2.version': No such file or directory  
→ stereo_calibration adb reboot
```

重启飞机即可。重启后，adb shell 可访问飞机。

2.3 检查双目图像

连接 PC 和 飞机，飞机上电，进入 repo 的 `/tools/hover_monitor` 文件夹，首先复制 `Raw8_VGA` 飞机双目标定教程里的 `stereo_cam_monitor.sh` 到 `repo/tools/hover_monitor`；其次复制 `image_parser.py` 到 `tools/hover_monitor/python`，取代原来文件。



最后，使飞机双目正对标定板（需要胶带固定好双目，见下图），cd 到 `repo/tools/hover_monitor`，终端（ZSH用户请切换到 **bash** 终端，因为ZSH不出图，原因是保见使用 **bash**编译的）执行

```
./stereo_cam_monitor.sh
```

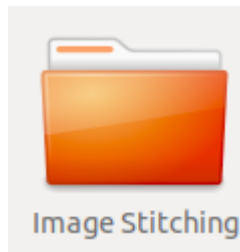
通过PC端查看双目图像的采集情况，使标定板位于图像中心并且占据最大画面，大概像下图(除了没有连接USB)



2.4 保存标定图像

当标定板位于图像中心并且占据最大画面时，保持飞机不动，把终端正在执行的 `./stereo_cam_monitor.sh` 任务用 `Ctrl+c` 停掉，系统自动保存该图像，名为 `origin.jpg` 位置在 `repo/tools/hover_monitor/python`

但该图像为上下粘在一起的，需要弄成左右才行，故进入 `image stitching` 文件夹



一定要用CLion打开该项目（直接用ubuntu终端会莫名出现opencv报错，可能是CMakeLists问题），执行

```
cd build
cmake .. && make
./TTest origin.jpg
```

如果报错，就把 `origin.jpg` 放到 `build` 文件夹下，然后再执行，就可以得到 `stereo.jpg`。

2.5 标定

1. 全程在Windows系统下，笔者是 win10；
2. 拷贝本说明同级目录下的yaml文件操作指导到windows系统并解压，以下操作都在[yaml文件操作指导]目录下进行



3. 把上一步骤生成的stereo.jpg图像放入pics文件夹
4. 清空result文件夹
5. 用终端执行stereo_calib
6. 在result文件夹中得到stereo.yaml文件

2.6 yaml进飞机

1. 飞机开机状态下 USB 连接电脑
2. `adb push stereo.yaml /mnt/persist/`
3. 标定完成。

3 飞机镜像

镜像：Fastboot_ZZ_IMG_B1_POSTCS3_V6.3.54，版本号即为 6.3.54

地址：<http://pan.zerozero.cn:88/index.php/apps/files/?dir=/Embedded/images/HoverCamera2/B1-Postcs3-jenkins&fileid=615361>

方法：

1. 用 USB 连接飞机和 PC；
2. PC 终端 `cd /Fastboot_ZZ_IMG_B1_POSTCS3_V6.3.54/out/fastboot_build/emmc`
3. PC 终端运行 `./fastboot-all.sh` (中间会提示输入密码，为电脑密码)
4. 完了飞机自动重启，如果起不来，手动重启。

4 飞机 *ipk*

网址：<http://ci.zerozero.cn:88/job/HC2Repo-Branch-feature-oademo/>

方法: 以 `HC2Pro_feature_oademo.release_95` 为例

1. 下载并解压 `HC2Pro_feature_oademo.release_95`
2. 终端下运行 `./install.sh` 之后重启生效。

5 飞机 *Captain*

编译方式同编译 repo 里的代码，但要提前注意切换到 Captain 的分支，例如

```
cd /home/xian/workspace/repo/src/drone_platform/Captain/
```

该目录可以查看分支

```
git log
git checkout dev_p2_oademo
```

如果要使用 master 则

```
git checkout master
```

编译 master 版本的 captain，则

```
cd repo && cd build
ninja Captain
```

编译出来的绿色可执行文件在 `/home/xian/workspace/repo/src/drone_platform/Captain/` 目录下，将其使用 `adb push` 即可推进飞机。

6 飞机 *FC*

FC 简称飞控，在 repo 根目录，不同于编译 Captain 和 Autopilot（在 `/repo/build/` 目录下编译），它直接可以在 repo 根目录执行

```
cd workspace/repo/           //repo根目录
make fc
make push -fc
```

注意：由于 fc 东西很多，不像 Captain 和 Autopilot 就推一个绿色可执行文件到飞机，所以推的时候要推很多东西。如果单推某个文件，飞控会出问题。具体推了什么，笔者没试过，建议看屏幕上的 log。

7 推结果进 *PC*

相当于刷飞机的 ipk。操作是：

[1] 关闭服务，否则推不进去

```
adb shell systemctl stop zz_pathplanning
```

[2] 推可执行文件

```
adb push
```

```
/home/xian/workspace/repo/src/drone_platform/vision/Autopilot/cmake/aarch64/AutoPilot /hover/PathPlanningNeo/
```

[3] 推so库

```
adb push
```

```
/home/xian/workspace/repo/src/drone_platform/vision/Autopilot/cmake/aarch64/libPathPlanning.so /hover/lib64/
```

[4] 推 CD_params.yml

```
adb push CD_params.yml /hover/PathPlanningNeo/
```

[5] 推 setting.yml

```
adb push setting.yml /hover/PathPlanningNeo/
```

就把东西推进了飞机相应位置

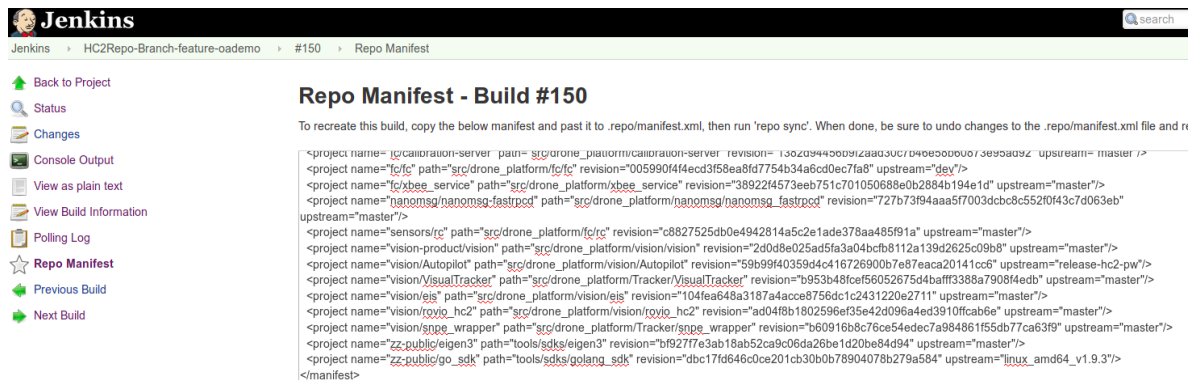
注意：

- 不要自己编译 Captain，因为Captain费机器，耗时；
- 不要自己推飞控，万一错了，飞机就废了。

7 Ci上查看构建ipk的版本号

提示：如果只是飞机快速起飞，测试，只需要镜像和构建好的ipk. 小节 6 就是自己编译 ipk。

【方法1】进入公司 Ci网址，然后点击构建的版本，例如查看 #150，然后点击 Repo Manifest 可以看到 upstream master 以及 upstream="release-hc2-pw" 等字样，这些就是版本号。



```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="captain" path="src/drone_platform/captain" revision="1.0.0" upstream="master" />
<project name="vision" path="src/drone_platform/vision" revision="2.0.0" upstream="master" />
<project name="autopilot" path="src/drone_platform/autopilot" revision="1.0.0" upstream="master" />
<project name="pathplanning" path="src/drone_platform/pathplanning" revision="1.0.0" upstream="master" />
<project name="sdk" path="src/drone_platform/sdk" revision="1.0.0" upstream="master" />
```

【方法2】进入本地PC，找到 Captain或 fc 或 AutoPilot文件目录，进入后 ZSH会提示用户在什么分支，然后 git log 一下就可以知道了。

常见问题

1 飞机无法起飞

检查：

- 是否装了新构建的 ipk，例如 121 版本；
- 检查是否推入双目标定文件；
- 电量是否不足；
- 新构建的版本是否要求地面起飞；
- sonar 是否更新

2 声呐更新

注意：

- 先确认飞机是 b3 还是 b2，因为二者声呐是不一样的，如果不知道，一般为 b3；
- 是否重刷了 ipk，有时重刷了 ipk，声呐也需要更新。

官网地址：http://ci.zerozero.cn:88/job/HC2_Sonar_Dev/1/

下载声呐，然后飞机 USB 链接电脑，执行

```
adb push Object.img /tmp //飞机的tmp目录，下次启动会删除该无用文件
adb shell systemctl stop zz_fc
adb shell
cd hover/tests/sensors/
./update_module_test sonar /tmp/Object.img
adb reboot
```

必须出现 `success` 字样才说明成功，例如

```
~/application/1.1.155/tests/sensors # ./update_module_test sonar
/tmp/Object.img
module = sonar
path= /tmp/Object.img
sonar board software version: 0, 0, 0 hardware version: 0, status: 0
first byte: 0x5a ====
54012 bytes will be update
start iap..
iap end: ret= 0.
success. //注意此行
```

3 无法进入飞机目录

主要发生在：重刷镜像+adb shell后。解决方案是：重刷镜像，再重刷 ipk 即可。

4 Repo记录上次编译错误

现象：

第一次编译：

```
cd workspace/repo/src/drone_platform/vision/  
git clone git@git.zerozero.cn:vision/Autopilot.git  
cd Autopilot  
git checkout release-hc2-pw  
git pull  
git checkout b12e7140f01  
  
# 新开终端  
cd workspace/repo/build  
ninja AutoPilot //OK, 编译通过
```

第二次编译：

```
# 将一套全新可正确编译的代码替换掉刚才 clone 下来的代码  
# 新开终端  
cd workspace/repo/build  
ninja AutoPilot //报错, 提示 clone下来的某些cpp文件缺失
```

解决方案：

第一步：清空编译

```
ninja clean
```

第二步：回到 `repo` 根目录

```
make
```

第三步：当 `make` 出现

```
[10/2268]xxxx.....std.o
```

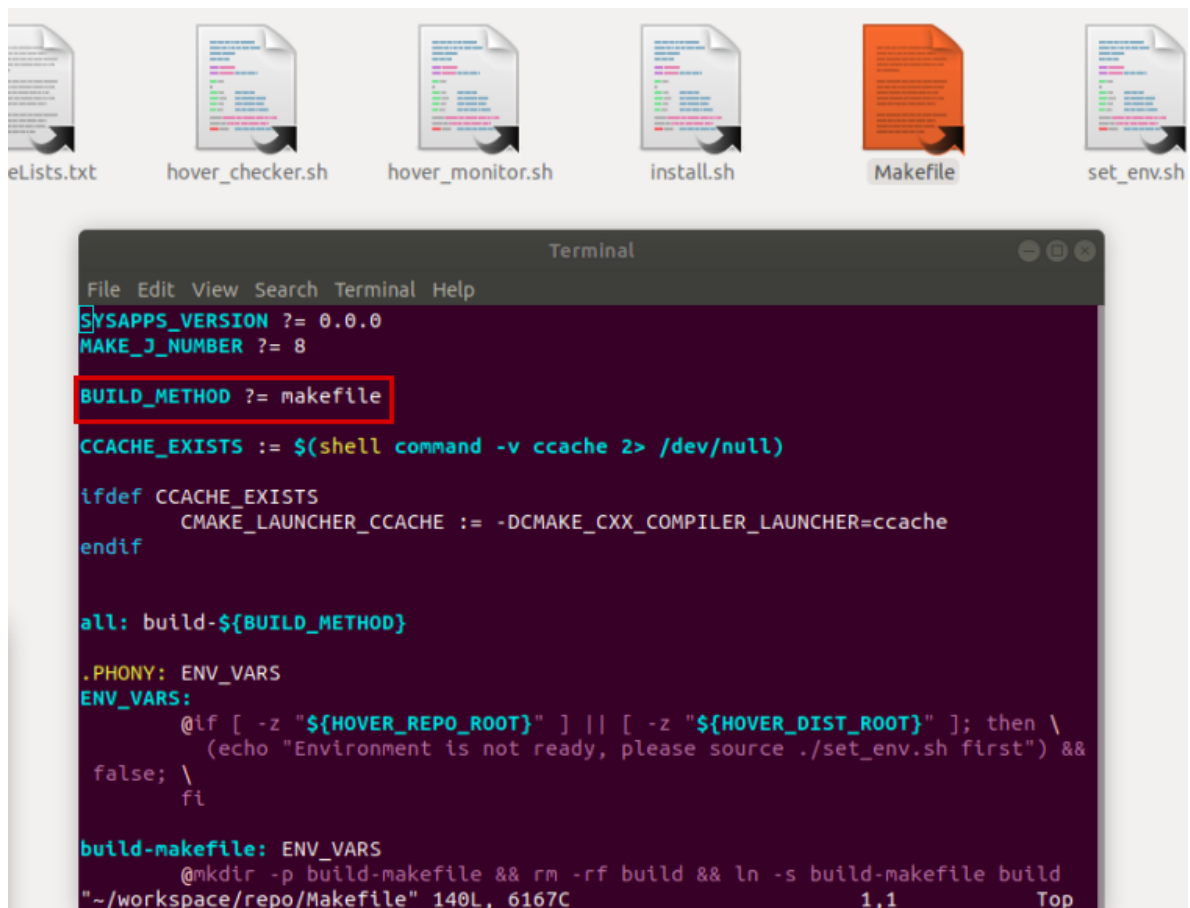
时，执行 `ctrl+C` 停止，然后重新编译

5 ninja切换到cmake编译

第一步：环境变量设置

```
gedit ~/.zshrc //也可以根据个人情况, 使用 geidt ~/.bashrc  
  
# added by Xian, if using ninja, just do  
# export BUILD_METHOD=ninja  
export BUILD_METHOD=makefile //注意此行
```

上面之所以是 `makefile`，是因为 `repo`根目录下的 `makefile` 文件（红框所示）



如果里面写着 `make`，则环境变量改为 `make` 即可，这个是CTO和嵌入式搞的。编译 `AutoPilot` 或 `stereo_cam`，需要到指定目录，例如

- AutoPilot:

`/home/xian/workspace/repo/build/src/drone_platform/vision/Autopilot`，然后执行 `make`，可执行文件在 `/home/xian/workspace/repo/build/src/drone_platform/vision/Autopilot/cmake/aarch64/` 目录下；

- stereo_cam： `/home/xian/workspace/repo/src/drone_platform/multimedia/camera_service`，然后执行 `make`，可执行文件在 `/home/xian/workspace/repo/src/drone_platform/multimedia/camera_service/src/tests/` 目录下。

6 标定视频流变为320*480

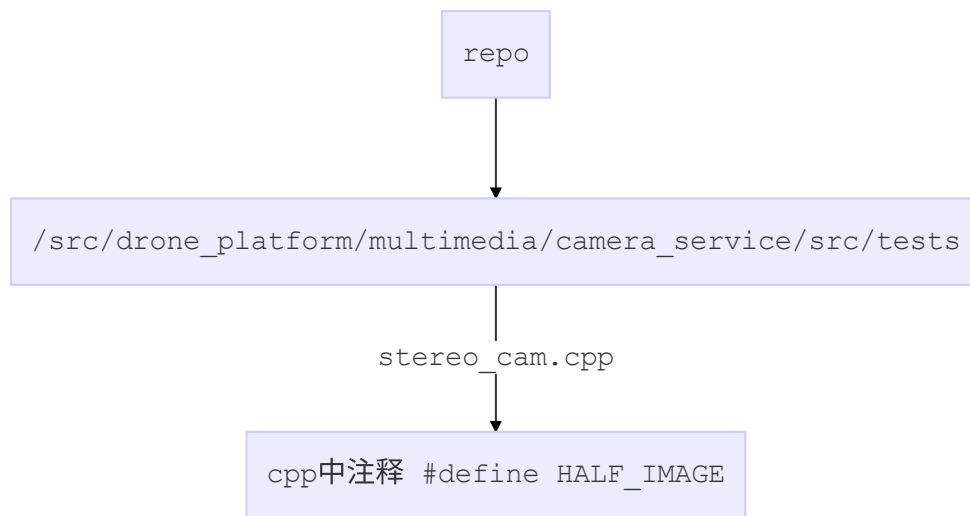
现象：标定时发现图片本应该是 640×960 的 `origin.jpg` 变成了 320×480 图片，导致标定生成的 `stereo.yaml` 参数缩小一半，例如 M1,M2矩阵的元素缩小一半

```
# 报错提示
→ hover_monitor git:(c8a04e8) x ./stereo_cam_monitor.sh --mode_ssh
Traceback (most recent call last):
  File "image_parser.py", line 33, in <module>
    img = bs.reshape(height, width) # reshape array to image shape
ValueError: cannot reshape array of size 153600 into shape (960,640)
```

原因：有人修改了 camera 底层代码的东西，导致双目数据源成为 320*480图片。

解决：需要修改repo中的源码，然后编译，推入飞机，才能改变输出图片的大小。

6.1 repo 中源文件修改



进

入 `/home/xian/workspace/repo/src/drone_platform/multimedia/camera_service/src/test`
s 用vi 打开 `stereo_cam.cpp` 文件，注释红框中的代码

```
#include <assert.h>

#include "opencv2/opencv.hpp"
#include <nn.h>
#include <reqrep.h>
#include <pubsub.h>
#include "ElementCamera.h"
#include <logging.h>
#include <zzlog.h>
#include "exposure_adjust.h"
#include <mutex>
#include <thread>

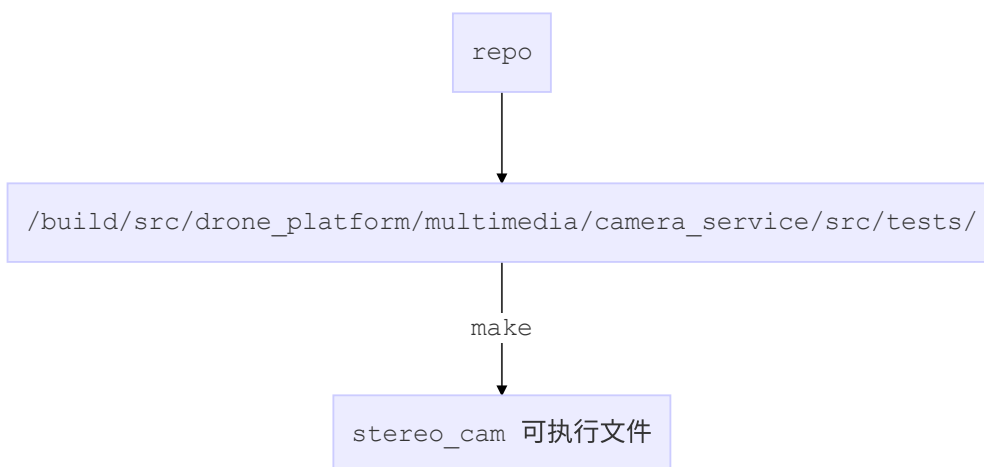
// #define HALF_IMAGE

#ifdef HALF_IMAGE
/** \brief Halfsamples an 1280x240 image to 640x240.
 *
 * @param pIn - Input image.
 * @param pOut - Output image (halfsampled).
 */
static void half_1280x240_to_640x240(uint8_t* pIn, uint8_t* pOut){
#define HALF_WIDTH 640
#define HALF_HEIGHT 240
```

注释后，双目算法将得到 640×480 图片；不注释，双目算法将得到 320×240的图片；

6.2 repo 编译

到 `/repo/build/src/drone_platform/multimedia/camera_service/` 目录下执行 `make`



6.3 结果推入飞机

位置：

```
adb shell
cd hover/stereo_cam
```

执行：

```
adb shell systemctl stop zz_*           //关闭飞机上所有服务，将来要重启
adb push stereo_cam /hover/stereo_cam    //注意 stereo_cam 的路径
```

6.4 image_parser解析

1. 回到 `repo/tools/hover_monitor` 目录，执行**第一个，使用大图**

```
# 使用大图
./stereo_cam_monitor.sh --mode_ssh    //产生 640*960大图

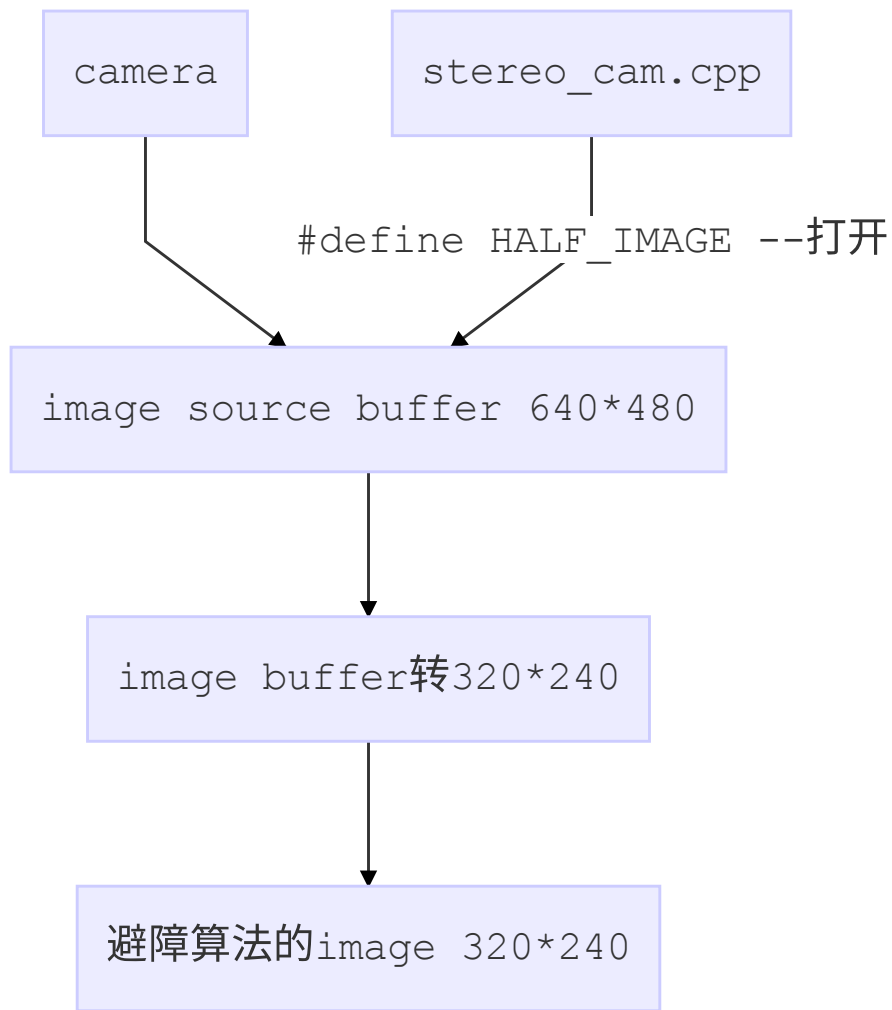
# 使用小图
./stereo_cam_monitor.sh --mode_usb    //产生 320*480小图
```

2. 在 `python` 目录下得到的就是 640×960上下排列的大图，该图就是用来标定用的 `origin.jpg`

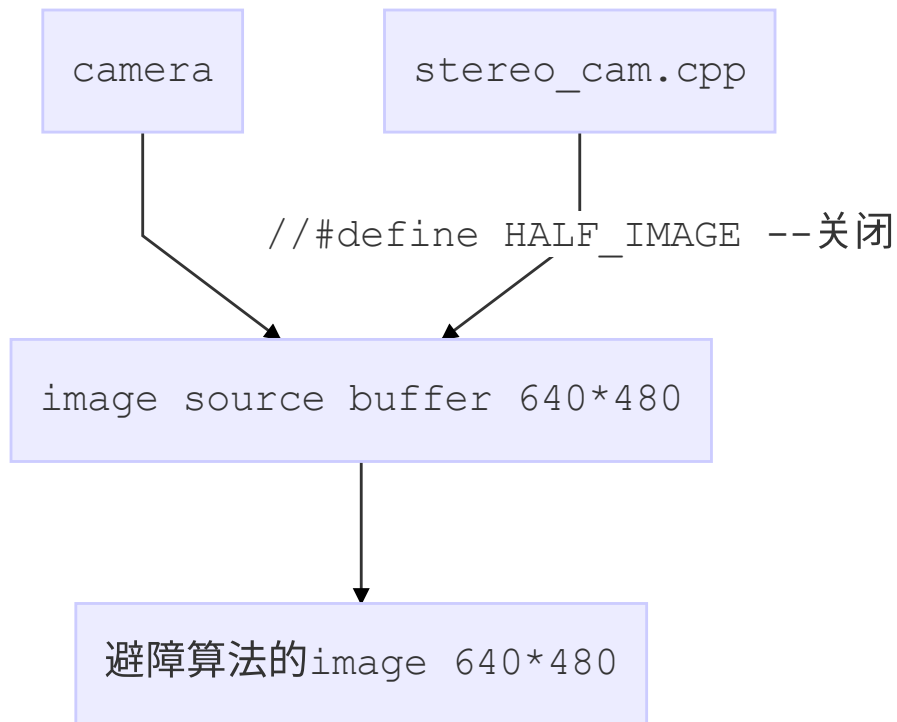
注意：标定完后记得把红框的注释打开，然后再次编译，推入飞机，因为双目屏障时用的是 320*240的小图。

6.5 补充：

【1】双目算法操作逻辑I



【2】双目算法逻辑II



6.6 验证

打开双目，飞机录 log，用 Autoreplay看。

6.7 总结

- 标定时，要注释掉 `#define HALF_IMAGE`；标定完了，记得打开；
- 改变飞机的视频流图片大小（给标定用），必须经过repo源文件修改和编译，产生可执行文件后推入飞机，重启生效，然后根据 `image_parser.py` 脚本得到 `origin.jpg`（上下排列）；
- 改变双目算法的视频流图片大小（避障代码用），基本同上，但用不到脚本。

7 飞机无法录log

解决方案：飞机刷入镜像和ipk后需要app矫正。不矫正，rovio没有数据，避障也不会运行。

2 编译整个repo,确定repo编译无误（如果出现）

```
GitLab: The project you were looking for could not be found.  
fatal: Could not read from remote repository.
```

```
Please make sure you have the correct access rights  
and the repository exists.
```

问嵌入式吕建峰主管，使用 `git config user.name`和`git config user.email`，将结果（git账户）给他即可开通权限。

1. 下载repo完成后, 创建 build 目录, cd build, 执行

```
cmake ..
```

报错：

```
Environment is not ready:  
  
1. make sure "source ./set_env.sh" already  
2. if still happened and using CLion, please do: "Tools -> CMake ->  
Reset Cache and Reload Project"
```

在build目录里执行

```
cd .. //返回build的上级目录  
source ./set_env.sh
```

提示：要安装ninja

```
[INFO] ninja-build is not installed, try to install automatically...  
TO MAKE SURE, please run apt-get update firstly!
```

执行

```
sudo apt-get update //一定要加sudo
```

再次执行

```
source ./set_env.sh
```

完了重新执行

```
cmake ..
```

出现 `Cofniguring done` 和 `Generating done`

2. 执行 `make`

报错：

```
/bin/sh: 1:  
/home/xian/workspace/hc2/tools/sdks/qrl_sdk/QRL_SDK/apq8096_drone_sdk/sysroots/x86_64-linux/usr/bin/aarch64-oe-linux/aarch64-oe-linux-g++: not found
```

咨询枫树，安装 ia32 库，

```
apt-get install ia32-libs
```

发现过时，根据终端提示

```
sudo apt-get install lib32ncurses5 lib32z1
```

同时，使用 `dpkg`

```
dpkg --print-foreign-architectures  
  
#显示  
i386
```

现在，重新删除 build，进入 repo 下载目录，

```
source ./set_env.sh  
mkdir build && cd build  
cmake ..  
make
```

3. make image(编译 ipk)

4. usb 连接飞机，make image/install (把编译好的 ipk 推入飞机)

PS: 3和4暂时用不到,以后需要推整个 ipk 时可以用这个方法

备用参考

单独编译避障并推入飞机

飞机需要提前安装好的环境:

1. 镜像: Fastboot_ZZ_IMG_B1_POSTCS3_V6.3.54

方法:

1. cd /Fastboot_ZZ_IMG_B1_POSTCS3_V6.3.54/out/fastboot_build/emmc
2. 终端下运行 ./fastboot-all.sh (中间会提示输入密码,为电脑密码)

2. ci上已构建好的ipk

方法: 以HC2Pro_feature_oademo.release_95为例

1. cd HC2Pro_feature_oademo.release_95
2. 终端下运行 ./install.sh

3. 编译修改的OA程序

cd repo/build

cd src/drone_platform/vision/Autopilot

make

make完以后会在当前目录下的cmake/aarch64下产生要推到飞机上的内容

连接飞机,

```
adb shell systemctl stop zz_pathplanning && adb push cmake/aarch64/AutoPilot  
/hover/PathPlanningNeo/ && adb push cmake/aarch64/libPathPlanning.so /hover/lib64/ && adb  
reboot
```

就把东西推进了飞机相应位置