天途
TTA VIATION

无人机图像处理与控制

目录

- 计算机视觉基础介绍
- 机载算力盒子简介与快速入门

计算机视觉基础介绍

天途
TTA VIATION

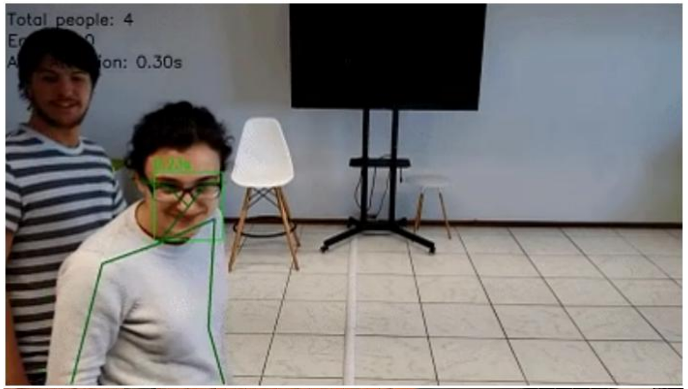
- 计算机视觉概述
- 计算机视觉典型任务
- AprilTag简介与简单使用

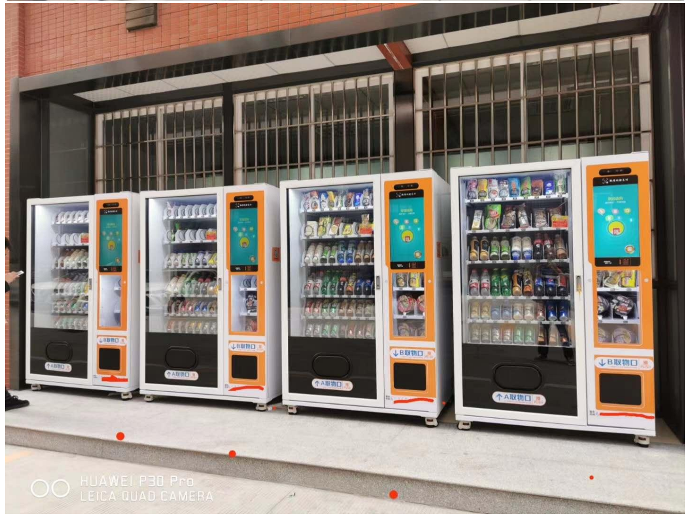
计算机视觉基础介绍

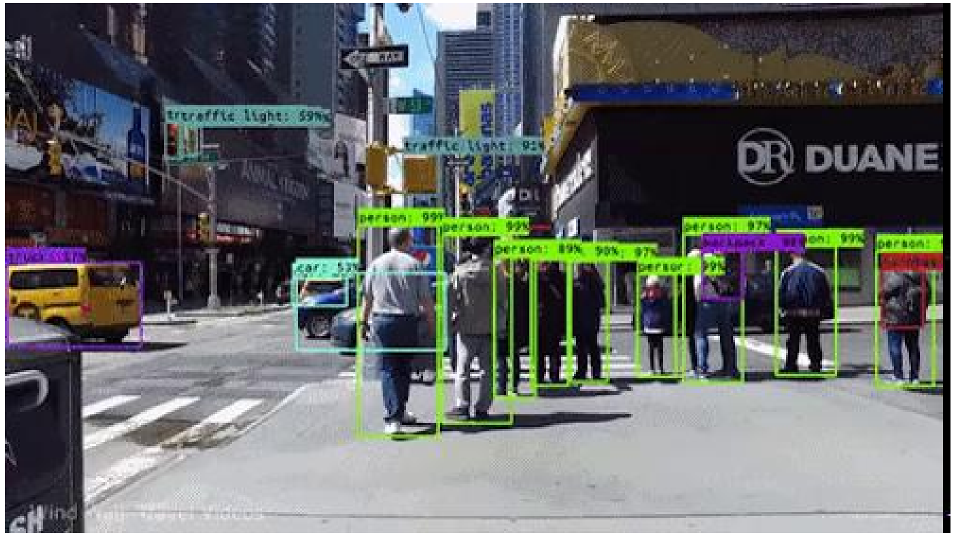
计算机视觉解决什么问题

给出一张二维图像，计算机视觉系统必须识别出图像中的对象及其特征，如形状、纹理、颜色、大小、空间排列等，从而尽可能完整地描述该图像。

计算机视觉行业应用







零售业

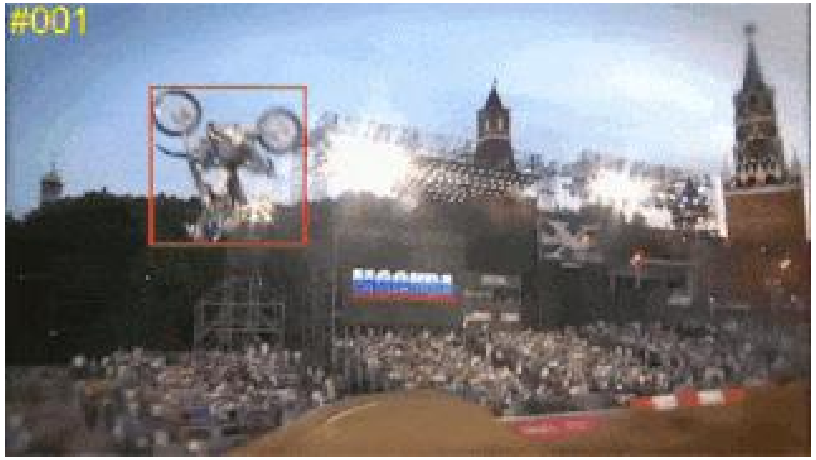
医疗行业

自动驾驶

计算机视觉典型任务

天途 TTA VIATION

图像分类

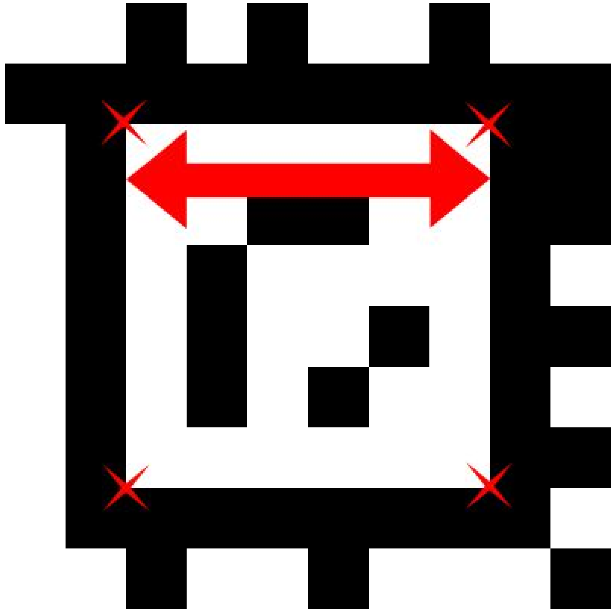


目标追踪

目标检测

实例分割

AprilTag简介与简单使用



AprilTag 在2011年由密歇根大学开发，是类似二维码编码方式生成的带有信息的方形图案。对方形内部黑白区域实现不同的布局，可以形成不同的AprilTag。

通过特定的算法，在摄像机拍摄的图像中可以精确地检测出 AprilTag，并且可以精确估计出摄像机坐标系相对于 AprilTag 坐标系的转换关系。常应用于无人机视觉定位等领域。

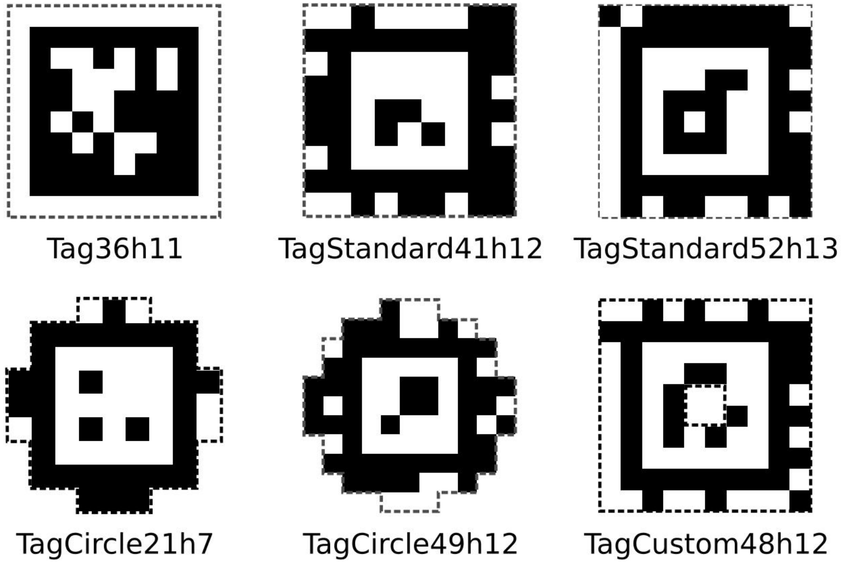
AprilTag是机器人研究中流行的视觉基准系统。该存储库包含AprilTag的最新版本AprilTag 3，改进了小标签检测率，提供了灵活的标签布局和姿态估计。

安装方法：

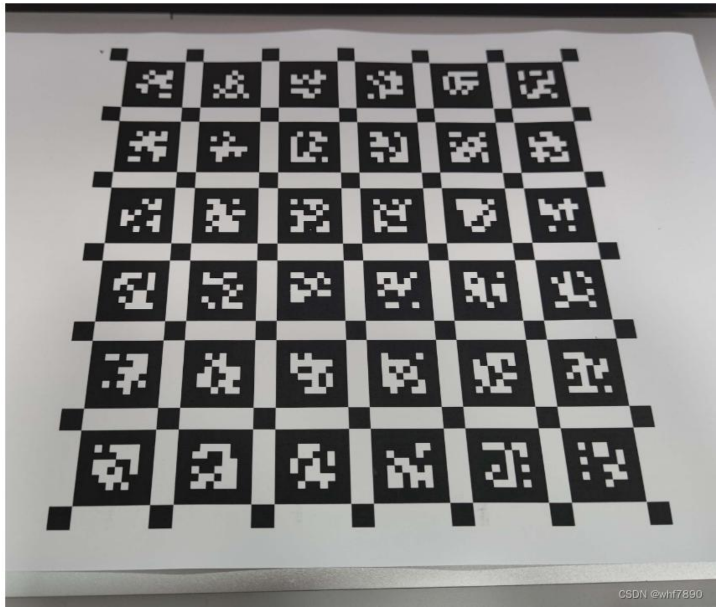
```
conda install scikit-build
pip install pupil-apriltags
```

参考链接：https://github.com/AprilRobotics/apriltag

AprilTag简介与简单使用



通过对AprilTag Marker的识别，可以确定相机的位姿(相对于Marker)。AprilTag除了常规的方形，还可以包含其它“奇形怪状”的样子。



一个普通的AprilTag大约长上面这个样子。

AprilTag

优点：

(1) BSD许可
(2) 更少的可调节参数
(3) 在长距离场景中表现依然较好
(4) 被NASA采用
(5) 更加灵活的marker设计(maker不一定是方形的)
(6) 计算量更少
(7) 支持Tag bundle，减少旋转歧义性
(8) 更多的误检测(默认参数)

参考链接：https://github.com/AprilRobotics/apriltag

AprilTag简介与简单使用

/camera/image_rect

话题发布消息类型为/sensor_msgs/Image，该消息包含了从相机采集到的图像数据。

/camera/camera_info

话题发布消息类型为/sensor_msgs/CameraInfo，该消息包含了相机的内参矩阵K和其他的一些标定参数。可以通过 camera_calibration（camera_calibration - ROS Wiki）或者kalibr（GitHub - ethz-asl/kalibr: The Kalibr visual-inertial calibration toolbox）获得标定参数。

tag.yaml文件中包含了用于检测的二维码信息。

settings.yaml文件中包含了apriltag算法核心配置。

/tf

话题包含了每个被检测到的二维码相对于相机的位置和方向的数据。只有在settings.yaml文件中将 publish_tf 设置为 true 才会发布。

/tag_detections

话题发布的内容和 /tf 一样，只是包含了一个自定义消息 tag ID，该消息主要用于一簇标签（tag bundles）的检测。

/tag_detections_image

话题发布的内容和 /camera/image_rect 内容一样，只是包含了标签绑定的内容（即在输出的图像上实时高亮显示标签的位置）。只有在continuous_detection.launch文件中，设置publish_tag_detections_image==true才会发布。

参考链接：https://github.com/AprilRobotics/apriltag

AprilTag简介与简单使用

天途 TTA VIATION

- **二维码的生成**

- 下载openmv（https://openmv.io/pages/download）这里选用的是linux版本
- 找到下载的源文件，赋予运行权限sudo chmod +x openmv-ide-linux-x86_64-2.6.5.run （这里要修改为自己下载的版本名称）
- 运行openmv./openmv-ide-linux-x86_64-2.6.5.run
- 然后选择 工具---机器视觉---ApirlTag生成器---TAG36H11，在弹出的对话框中选择需要的标签数量和对应的ID【这里选择的是TAG36H11，在**apriltag_ros/config/settings.yaml**文件中也要设置成对应的标签名称，否则算法将无法识别】

参考链接：https://github.com/AprilRobotics/apriltag

AprilTag简介与简单使用

➢ **启动camera_calibration**

roslaunch usb_cam usb_cam-test.launch

➢ **启动camera_calibration**

- 校准相机

rosrun camera_calibration cameracalibrator.py --size 11x9 --square 0.100 image:=/usb_cam/image_raw camera:=/usb_cam

- size 为棋盘标定板的大小
- square为棋盘的边长
- image为相机发布的图像信息（此处需要根据实际情况修改为自己相机发布的话题，可以通过 rostopic list 查看）
- camera 为相机发布的相机消息，同样，根据实际情况修改为自己的相机

AprilTag简介与简单使用

➢ 启动**apriltag_ros**

roslaunch apriltag_ros continuous_detection.launch
需要注意的是，在apriltag_ros/launch/continuous_detection.launch 文件下，
<arg name="camera_name" default="/usb_cam" />
<arg name="camera_frame" default="usb_cam" />
<arg name="image_topic" default="image_raw" />
将上面三个参数改为自己相机参数。

在apriltag_ros/config/tags.yaml 文件中修改为自己的标签设置。

standalone_tags:
[{id: 0, size: 0.1},{id: 1, size: 0.1},{id: 2, size: 0.1},{id: 3, size: 0.1},{id: 4, size: 0.1},{id: 5, size: 0.1},{id: 6, size: 0.1},{id: 7, size: 0.1},{id: 8, size: 0.1},{id: 9, size: 0.1},]

上面为此次测试时用的标签参数，包含了标签的**ID**和每个标签的大小**0.05m**

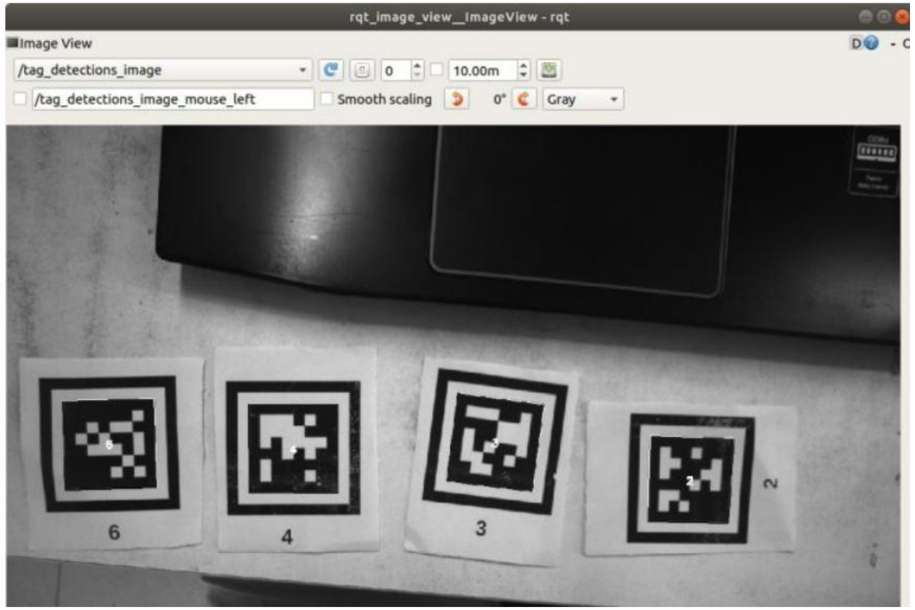
AprilTag简介与简单使用

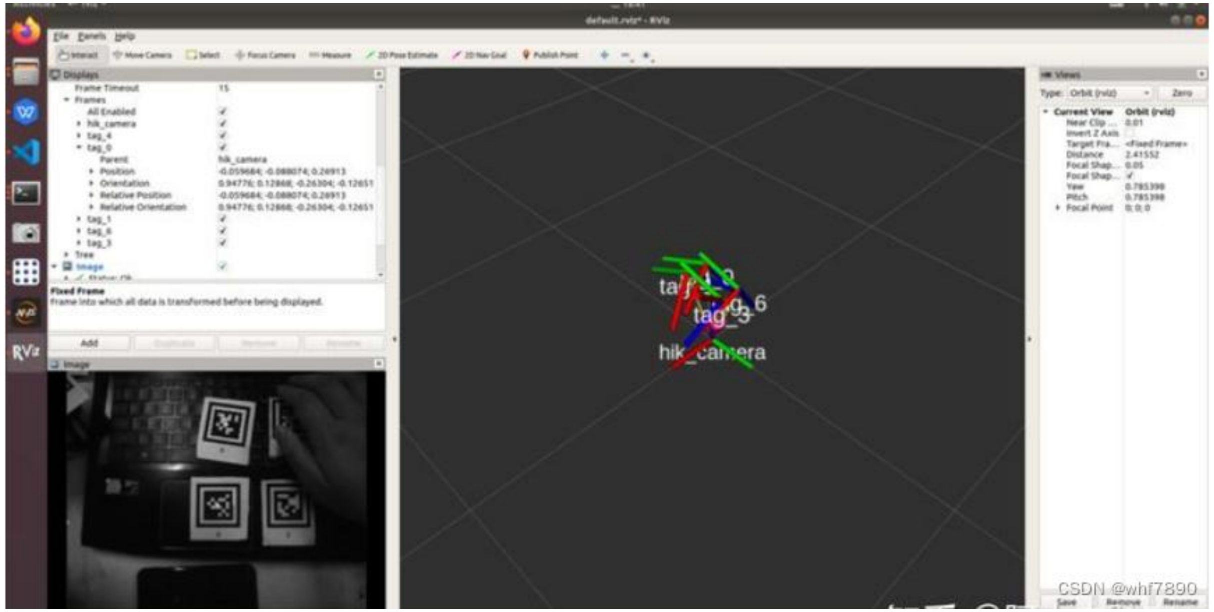
➢ **启动rqt_image_view**

话题选择 /tag_detectios_image，就可以观看到带有标签高亮的图像

➢ **启动rviz**

将固定坐标设置为 相机的坐标系（这里为 /usb_cam），并且添加TF模块，就可以实时看见每个tag相对于相机的位置和方向.
然后在ros下，建立工作空间，编译通过后，发布话题查看内容就好了。





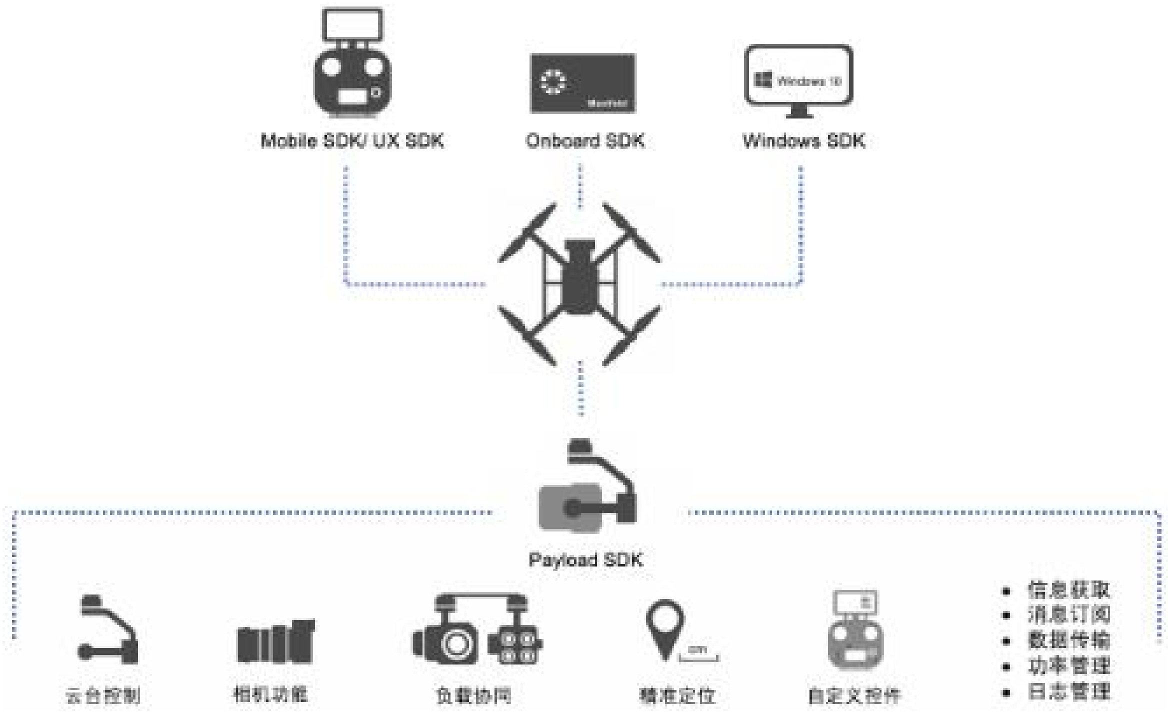
机载算力盒子简介与快速入门

天途
TTA VIATION

- PSDK简介
- 机载算力盒子快速入门
- 获取DJI无人机视觉感知系统的码流数据

➢ PSDK简介

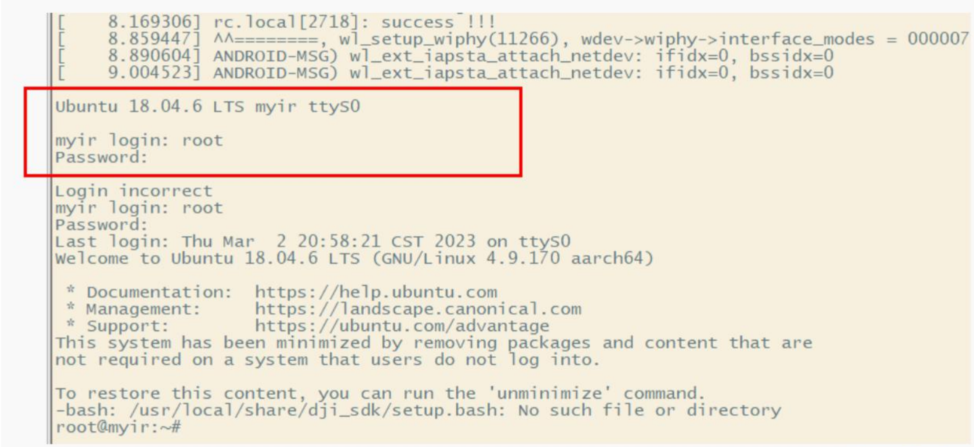
DJI 为支持开发者开发出可挂载在DJI 无人机上的负载设备，提供了**Payload SDK (即PSDK)** 、X-Port 标准云台和SkyPort 转接环，方便开发者利用DJI无人机上如电源、通讯链路及状态信息等资源，开发出可挂载在DJI 无人机上的负载设备。



机载算力盒子快速入门

➢ 系统登录

串口连接算力盒子，可以看到如图信息。登录系统：



用户名：root
密码：123456

➢ 程序烧录

1、 下载镜像文件

链接: https://pan.baidu.com/s/1gfdQZUe0ir-lPirm3XEotw

提取码： 9pty

2、 烧录

A、 SD 卡插入算力盒子，格式化： mkfs.ext4 /dev/ mmcblk1

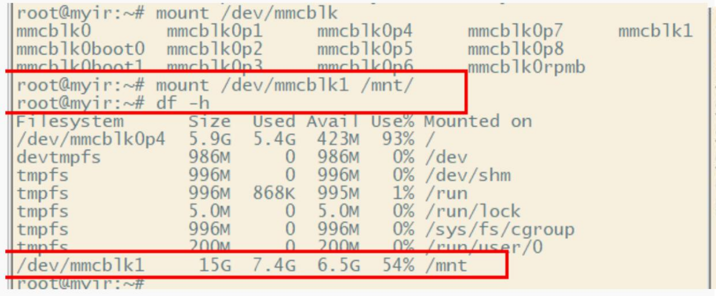
B、 拔出 SD 卡，拷贝镜像到 SD 卡中。

C、 SD 卡插入算力盒子，重新启动

D、 挂载： mount /dev/mmcblk1 /mnt/，烧录

指令： dd of=/dev/mmcblk0 if=/mnt/backup.img bs=1MiB status=progress

E、 烧录完成，重启



机载算力盒子快速入门

天途
TTA VIATION

➢ Psdk 程序运行

1、cd /root/tta_ros
2、执行命令：
sudo ./dji_sdk_demo_linux_cxx

```
root@myir:~/tta_ros# sudo ./dji_sdk_demo_linux_cxx
[0.006][core]-[Info]-[DjiCore_Init:96) Payload SDK Version : V3.4.0-beta.0-build.1743
[0.139][adapter]-[Info]-[DjiAccessAdapter_Init:180) Identify aircraft series is Mavic 3 Series
[0.139][adapter]-[Info]-[DjiAccessAdapter_Init:198) Identify mount position type is Extension Port Type
[2.157][adapter]-[Info]-[DjiPayloadNegotiate_Init:185) Waiting payload negotiate finish.
[3.157][adapter]-[Info]-[DjiPayloadNegotiate_Init:189) No need wait negotiate finished
[5.208][adapter]-[Info]-[DjiPayloadNegotiate_Init:185) Waiting payload negotiate finish.
[6.208][adapter]-[Info]-[DjiPayloadNegotiate_Init:189) No need wait negotiate finished
[6.209][core]-[Info]-[DjiIdentityVerify_UpdatePolicy:445) Updating dji sdk policy file...
[7.209][core]-[Info]-[DjiIdentityVerify_UpdatePolicy:448) Update dji sdk policy file successfully
[7.239][core]-[Info]-[DjiCore_Init:164) Identify AircraftType = Mavic 3 Enterprise, MountPosition = Extension Port, SdkAdapterType = None
[7.244][core]-[Info]-[DjiCore_ApplicationStart:231) Start dji sdk application
[7.244][user]-[Info]-[DjiUser_ApplicationStart:260) Application start.
[10.276][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[10.276][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 10273 microsecond 10273000.
[10.277][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993765 0.000452 0.037764 -0.104901.
[10.277][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.40 yaw = -12.07.

[10.734][user]-[Info]-[Dji_FcSubscriptionStartService:305) dji system module init success!!
[10.734][flight]-[Info]-[DjiFlightController_RegisterLinkerObj_M3:111) Init mavic3 enterprise series flight controller linker successfully.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 11279 microsecond 11279000.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993791 0.000412 0.037755 -0.104659.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.41 yaw = -12.04.

[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 12285 microsecond 12285000.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993790 0.000587 0.037832 -0.104641.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.32 roll = -0.39 yaw = -12.04.

[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 13296 microsecond 13296000.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993779 0.000621 0.037768 -0.104765.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.38 yaw = -12.05.

^C
```

➢ ROS 节点程序运行

1、 ros 节点程序目录：
/root/catkin_ws/
2、 source ./devel/setup.bash
3、 启动 ROS 环境

```
root@myir:~/catkin_ws#
root@myir:~/catkin_ws#
root@myir:~/catkin_ws#
root@myir:~/catkin_ws# roscore
... logging to /root/.ros/log/7962da78-b8fb-11ed-ac26-babeb8e62f06/roslaunch-myir-7595.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://myir:38607/
ros_comm version 1.14.13


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [8077]
ROS_MASTER_URI=http://myir:11311/

setting /run_id to 7962da78-b8fb-11ed-ac26-babeb8e62f06
process[rosout-1]: started with pid [8242]
started core service [/rosout]
```

机载算力盒子快速入门

➢ Psdk 程序运行

1、cd /root/tta_ros
2、执行命令：
sudo ./dji_sdk_demo_linux_cxx

```
root@myir:~/tta_ros# sudo ./dji_sdk_demo_linux_cxx
[0.006][core]-[Info]-[DjiCore_Init:96) Payload SDK Version : V3.4.0-beta.0-build.1743
[0.139][adapter]-[Info]-[DjiAccessAdapter_Init:180) Identify aircraft series is Mavic 3 Series
[0.139][adapter]-[Info]-[DjiAccessAdapter_Init:198) Identify mount position type is Extension Port Type
[2.157][adapter]-[Info]-[DjiPayloadNegotiate_Init:185) Waiting payload negotiate finish.
[3.157][adapter]-[Info]-[DjiPayloadNegotiate_Init:189) No need wait negotiate finished
[5.208][adapter]-[Info]-[DjiPayloadNegotiate_Init:185) Waiting payload negotiate finish.
[6.208][adapter]-[Info]-[DjiPayloadNegotiate_Init:189) No need wait negotiate finished
[6.209][core]-[Info]-[DjiIdentityVerify_UpdatePolicy:445) Updating dji sdk policy file...
[7.209][core]-[Info]-[DjiIdentityVerify_UpdatePolicy:448) Update dji sdk policy file successfully
[7.239][core]-[Info]-[DjiCore_Init:164) Identify AircraftType = Mavic 3 Enterprise, MountPosition = Extension Port, SdkAdapterType = None
[7.244][core]-[Info]-[DjiCore_ApplicationStart:231) Start dji sdk application
[7.244][user]-[Info]-[DjiUser_ApplicationStart:260) Application start.
[10.276][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[10.276][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 10273 microsecond 10273000.
[10.277][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993765 0.000452 0.037764 -0.104901.
[10.277][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.40 yaw = -12.07.

[10.734][user]-[Info]-[Dji_FcSubscriptionStartService:305) dji system module init success!!
[10.734][flight]-[Info]-[DjiFlightController_RegisterLinkerObj_M3:111) Init mavic3 enterprise series flight controller linker successfully.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 11279 microsecond 11279000.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993791 0.000412 0.037755 -0.104659.
[11.279][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.41 yaw = -12.04.

[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 12285 microsecond 12285000.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993790 0.000587 0.037832 -0.104641.
[12.285][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.32 roll = -0.39 yaw = -12.04.

[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:626) receive quaternion data.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:627) timestamp: millisecond 13296 microsecond 13296000.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:629) quaternion: 0.993779 0.000621 0.037768 -0.104765.
[13.296][user]-[Info]-[Dji_FcSubscriptionReceiveQuaternionCallback:632) euler angles: pitch = 4.31 roll = -0.38 yaw = -12.05.

^C
```

➢ ROS 节点程序运行

1、 ros 节点程序目录：/root/catkin_ws/
2、 source ./devel/setup.bash
3、 启动 ROS 环境
4、 换另一个中断，执行 ROS 节点程序。飞机数据节点：rosrun ttauav_node uavdata

```
root@myir:~/catkin_ws#
root@myir:~/catkin_ws#
root@myir:~/catkin_ws# roscore
... logging to /root/.ros/log/7962da78-b8fb-11ed-ac26-babeb8e62f06/roslau
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://myir:38607/
ros_comm version 1.14.13


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [8077]
ROS_MASTER_URI=http://myir:11311/

setting /run_id to 7962da78-b8fb-11ed-ac26-babeb8e62f06
process[rosout-1]: started with pid [8242]
started core service [/rosout]
```

```
root@myir:~/catkin_ws# rosrun ttauav_node uavdata
new -----------------------------------DataPortKeeperThread
get msgid 8202 form FC
get msgid 8202 form FC
get msgid 8202 form FC
get msgid 8194 form FC
get msgid 8201 form FC
get msgid 8202 form FC
get msgid 8202 form FC
get msgid 8202 form FC
get msgid 8201 form FC
get msgid 8202 form FC
get msgid 8202 form FC
get msgid 8202 form FC
```

如图，表示正确接收到 psdk 数据。

无人机上报的数据如下图：

```
# positon in WGS84
float64 latit
float64 longi
float32 altit

float32 velN
float32 velE
float32 velD

float32 atti_pitch
float32 atti_roll
float32 atti_yaw

float32 gyro_pitch
float32 gyro_roll
float32 gyro_yaw

float32 accN
float32 accE
float32 accD

float32[] quat
```

机载算力盒子快速入门

➢ **Psdk 程序运行**

1、cd /root/tta_ros

2、执行命令：

sudo ./dji_sdk_demo_linux_cxx

```
root@myir:~/catkin_ws/src/raspicam_node/src# rosrun ttauav_node listener
[ INFO] [1677762201.711624581]: I heard:[40.195992]
[ INFO] [1677762201.722732833]: I heard:[116.168511]
[ INFO] [1677762201.723000333]: I heard:[0.995497]
[ INFO] [1677762201.723063541]: I heard:[-0.000491]
[ INFO] [1677762201.723117916]: I heard:[0.037417]
[ INFO] [1677762201.723168874]: I heard:[-0.087099]
[ INFO] [1677762201.810789675]: I heard:[40.195992]
[ INFO] [1677762201.810960216]: I heard:[116.168511]
[ INFO] [1677762201.811054966]: I heard:[0.995495]
[ INFO] [1677762201.811141258]: I heard:[-0.000473]
[ INFO] [1677762201.811225133]: I heard:[0.037423]
```

如图所示，即为数据监听成功。

➢ **飞行控制**

启动飞行控制服务：

```
root@myir:~/catkin_ws#
root@myir:~/catkin_ws# rosrun ttauav_node service
new ---------------------------------------DataPortKeeperThread
[ INFO] [1677762717.553434912]: Start takeoffOrLanding server
[ INFO] [1677762717.569791163]: Start flight server
```

飞行控制参数说明：

```
float32[] offset

float32 targetYaw

float32 yawThreshold
float32 posThreshold

---

int8 ack
```

Offset[0]: 北向移动距离，单位米
Offset[1]: 东向移动距离，单位米
Offset[2]: 向上移动距离，单位米
TargetYaw：目标航向值，单位度
yawThreshold：航向精度阈值
posThreshold：位置精度阈值

机载算力盒子快速入门

天途
TTA VIATION

➤ **代码**

所有ROS程序源码， 均在 /root/catkin_ws/src/raspicam_node 下

```
tta@ubuntu: /workspace/catkin_ws/src/raspicam_node$ tree

├─CHANGELOG.rst
├─CMakeLists.txt
├─include
├─launch
│  └─uavdata.launch
├─LICENSE
├─msg
│  └─uavdata.msg
├─package.xml
├─pipline.py
├─README.md
├─reconfigure_raspicam_node.png
├─src
│  ├─flightService.cpp
│  ├─listener.cpp
│  ├─proxy_src
│  │  ├─Buffer.cpp
│  │  ├─Buffer.h
│  │  ├─BufferItem.cpp
│  │  ├─BufferItem.h
│  │  ├─DataPort.cpp
│  │  ├─DataPort.h
│  │  ├─DataPortKeeperThread.cpp
│  │  ├─DataPortKeeperThread.h
│  │  ├─DataPortReadThread.cpp
│  │  ├─DataPortReadThread.h
│  │  ├─DataPortSendThread.cpp
│  │  ├─DataPortSendThread.h
│  │  ├─Lock.cpp
│  │  ├─Lock.h
│  │  ├─Log.cpp
│  │  ├─Log.h
│  │  ├─Port.cpp
│  │  ├─Port.h
│  │  ├─SerialDataPort.cpp
│  │  ├─SerialDataPort.h
│  │  ├─SerialPort.cpp
│  │  ├─SerialPort.h
│  │  ├─SocketUtils.cpp
│  │  ├─SocketUtils.h
│  │  ├─TcpDataPort.cpp
│  │  ├─TcpDataPort.h
│  │  ├─TcpPort.cpp
│  │  ├─TcpPort.h
│  │  ├─Thread.cpp
│  │  ├─Thread.h
│  │  ├─UdpDataPort.cpp
│  │  ├─UdpDataPort.h
```

```
│  ├─ttauav_node.cpp
│  ├─uavData.cpp
│  ├─uavData.h
│  ├─uavpos_node.cpp
│  └─utils
│     ├─cJSON.c
│     ├─cJSON.h
│     ├─CJsonObject.cpp
│     ├─CJsonObject.hpp
│     ├─CryptUtils.cpp
│     ├─CryptUtils.h
│     ├─diskInfo.cpp
│     ├─diskInfo.h
│     ├─FileUtils.cpp
│     ├─FileUtils.h
│     ├─misc.cpp
│     ├─misc.h
│     ├─public_utils.cpp
│     ├─public_utils.h
│     ├─ttalinkUtils.cpp
│     └─ttalinkUtils.h
├─srv
│  ├─flight.srv
│  └─takeoffOrLanding.srv
└─tools
```

获取DJI无人机视觉系统的码流数据

- **概述**

DJI 开放了无人机视觉感知系统，开发者通过使用 PSDK 提供的 API 即可获取 DJI 无人机视觉感知系统的码流数据，并生成时差图以及点云图，结合图像识别算法，开发出满足特定使用场景需求的应用程序。

- **基础概念**

相机码流

为满足开发者使用 PSDK 开发的应用程序对获取相机码流的功能，PSDK 提供了获取相机码流的功能，支持获取 FPV 相机或获取 1 号云台相机 H.264 码流和 RGB 图像。

分辨率和帧频

PSDK 支持开发者获取 M300 RTK 无人机上 1 号云台上相机的码流，开发者或用户可根据实际的使用需求挂载不同型号的相机，根据相机的型号以及相机的工作模式，指定帧速率，获取所需的码流。

参考链接 https://developer.dji.com/doc/payload-sdk-tutorial/cn/function-set/advanced-function/liveview.html#%E4%BD%BF%E7%94%A8%E8%8E%B7%E5%8F%96%E7%9B%B8%E6%9C%BA-rgb-%E5%9B%BE%E5%83%8F%E7%9A%84%E5%8A%9F%E8%83%BD

获取DJI无人机视觉系统的码流数据

➢ **相机 H.264 码流**

1. 使用获取相机 H.264 码流的功能前，请开发者根据实际的使用需要先实现liveViewSampleCb函数，用于获取并处理相机 H.264 码流。

2. 调用startH264Stream()接口，指定所需获取码流的相机、接收相机 H.264 码流的回调函数和用户信息。

3. 开启无人机和用户负载设备，运行使用基于 PSDK 开发的应用程序，此时无人机将会向用户负载设备推送 H.264 码流。

4. 用户负载设备接收到 H.264 码流的数据后，将触发（作为入参传入开发者设置的回调函数中）基于 PSDK 开发的应用程序。

5. 开发者根据实际需求设计的函数liveViewSampleCb在获取相机 H.264 码流后，将对所获得的 H.264 码流执行存储、解码及转发等相应的操作。

参考链接 https://developer.dji.com/doc/payload-sdk-tutorial/cn/function-set/advanced-function/liveview.html#%E4%BD%BF%E7%94%A8%E8%8E%B7%E5%8F%96%E7%9B%B8%E6%9C%BA-rgb-%E5%9B%BE%E5%83%8F%E7%9A%84%E5%8A%9F%E8%83%BD

获取DJI无人机视觉系统的码流数据

开始获取相机 H.264 码流

```
LiveView::LiveViewErrCode startH264Stream(LiveView::LiveViewCameraPosition pos,
H264Callback cb, void *userData);
```

停止获取相机 H.264 码流

```
LiveView::LiveViewErrCode stopH264Stream(LiveView::LiveViewCameraPosition pos);
```

保存或处理相机 H.264 码流

```
typedef void (*H264Callback)(uint8_t* buf, int bufLen, void* userData);
```

参考链接 https://developer.dji.com/doc/payload-sdk-tutorial/cn/function-set/advanced-function/liveview.html#%E4%BD%BF%E7%94%A8%E8%8E%B7%E5%8F%96%E7%9B%B8%E6%9C%BA-rgb-%E5%9B%BE%E5%83%8F%E7%9A%84%E5%8A%9F%E8%83%BD

使用获取相机 RGB 图像的功能

在主线程中以轮询的方式获取 RGB 图像。

1. 创建主线程

调用vehicle->advancedSensing->startFPVCameraStream()接口创建一个线程，用于读取相机原始的码流并解码成图像。

2. 检查码流状态

开发者在主循环中，需调用vehicle->advancedSensing->newFPVCameraImageIsReady()接口，检查相机码流的状态，若相机中有可用的码流，则调用vehicle->advancedSensing->getFPVCameraImage(fpvImage) 获取该图像。
说明： 若开发者安装了 OpenCV 库，则可通过show_rgb函数调用cv::imshow()接口显示解码后的 RGB 图像。

3. 销毁线程

调用vehicle->advancedSensing->stopFPVCameraStream() 接口，断开与相机的链接，销毁读取相机码流的线程。

以回调函数的方式获取 RGB 图像

1. 创建获取相机码流的线程

调用vehicle->advancedSensing->startFPVCameraStream(&show_rgb, NULL)接口，创建获取相机码流的线程，同时在该接口中注册回调函数show_rgb，用于处理接收到的码流。

2. 销毁线程

调用vehicle->advancedSensing->stopFPVCameraStream()接口，断开与相机的连接，销毁读取相机码流的线程。

使用获取相机 RGB 图像的功能

以回调函数的方式获取 RGB 图像

1. 创建获取相机码流的线程

调用vehicle->advancedSensing->startFPVCameraStream(&show_rgb, NULL)接口，创建获取相机码流的线程，同时在该接口中注册回调函数show_rgb，用于处理接收到的码流。

2. 销毁线程

调用vehicle->advancedSensing->stopFPVCameraStream()接口，断开与相机的连接，销毁读取相机码流的线程。

Q&A