



无人机赛课第3讲

--仿真环境

李溢

联系方式: li-yi20@mails.tsinghua.edu.cn

时间: 2021年10月20日





目 录

- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结



目 录

- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结



1 仿真界面简介

• 环境依赖

- 操作系统: **Ubuntu 18.04**
- 相关软件:
 - **ROS Melodic**: 机器人操作系统;
 - **Gazebo 9.0.0**: 三维机器人仿真平台, 能够模拟室内/室外等各种复杂环境中的机器人;
 - **PX4 v1.9.2**: 自动驾驶仪固件, 可用于驱动无人机;
 - **MAVLink**: 消息传输协议, 用于地面控制终端 (地面站) 与无人机之间进行通信。





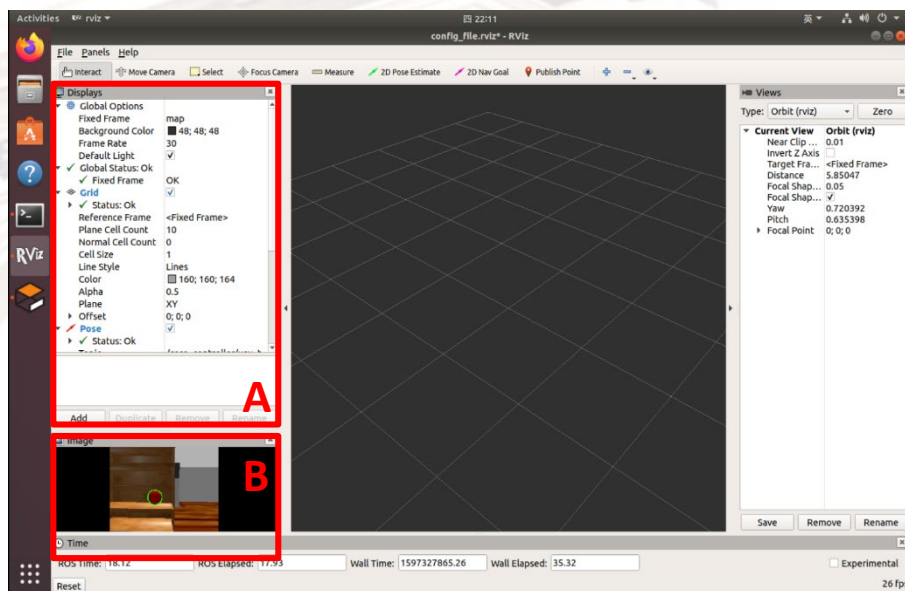
1 仿真界面简介

• 认识仿真界面

- 配置好仿真环境后，可以运行如下命令，先认识仿真界面：

```
roslaunch uav_sim arena_test_py.launch
```

- 该命令会启动Rviz和Gazebo两个窗口。
- Rviz为ROS自带的图形化工具，下图中 A 区域为订阅的一些topic信息，B区域为image类型的消息，下图中是对无人机摄像头拍摄图片处理后的图像（比如检测红色的球）。

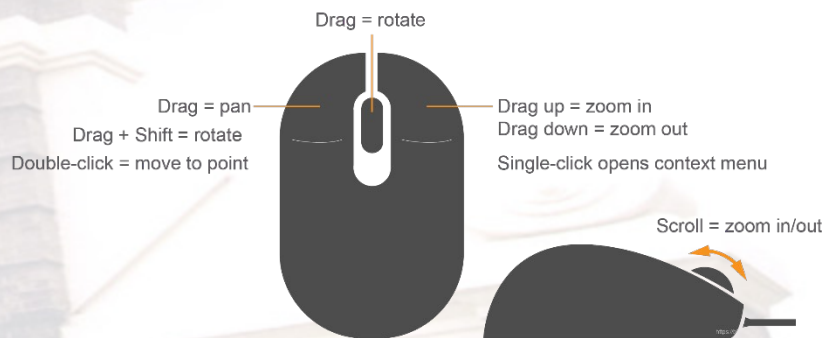
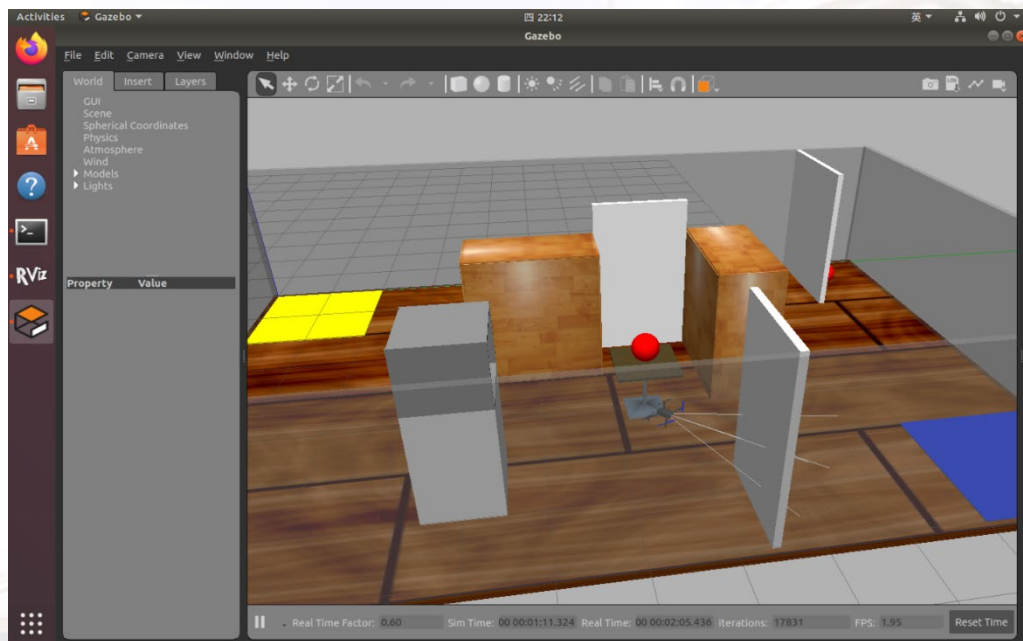




1 仿真界面简介

• 认识仿真界面

- 运行仿真例程后，会启动Rviz和Gazebo两个窗口。
- Gazebo会显示模拟的仿真场景与无人机飞行状态，可以进行拖动以转换视角。
- 下图中黄色区域为起飞点，蓝色区域为降落点。





目 录

- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结

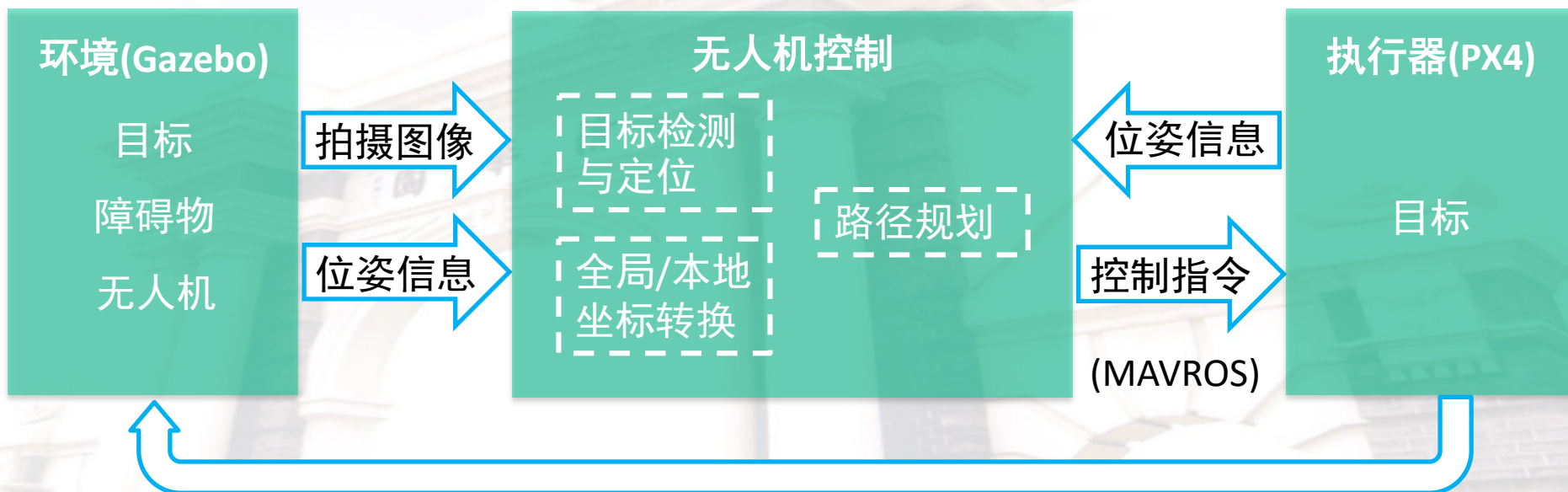


2 飞行控制流程

• 基本控制流程

– 无人机飞行流程如下图所示：

- Gazebo完成物理场景的模拟，通过发布话题的方式输出场景中各种物体的状态信息，以及无人机的位姿信息和拍摄的图像等；
- 无人机控制程序需要根据收集的数据进行信息融合并设计控制算法；
- 控制程序借助于MAVROS向无人机发送飞行指令(但是在仿真环境中，这一部分已经封装为tello的控制接口)，由PX4完成指令的执行。





目 录

- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结



2 数据采集

• 数据采集

- 无人机的状态和传感器数据发布在一些话题上：
 - **/tello/states**: 无人机的位置和姿态信息;
 - **/iris/usb_cam/** 下的几个话题: 包括相机参数(/iris/usb_cam/camera_info) 和无人机摄像头原始图像(/iris/usb_cam/image_raw)等。

ROS命令	作用
rostopic list	列出所有话题
rostopic info <topic_name>	获取话题信息, 包括消息类型、发布者和订阅者
rostopic type <topic_name>	获取消息类型
rosmmsg show <message_type>	获取消息格式

- 要了解消息格式中各项参数的详细含义可以参考官方文档 (http://docs.ros.org/melodic/api/gazebo_msgs/html/msg/ModelState.html)对于不同类型的消息只需要修改对应的关键字即可。



2 数据采集——位姿信息

• 数据采集——位姿信息

- Header: 包含了序列号、时间戳等信息;
- Pose: 包含了position(三维坐标)和orientation(四元数);

```
lee@drone:~$ rostopic info /tello/states
Type: geometry_msgs/PoseStamped

Publishers:
 * /env_util (http://drone:36747/)

Subscribers: None

lee@drone:~$ rosmmsg show geometry_msgs/PoseStamped
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
lee@drone:~$
```

话题/tello/states的消息格式

```
lee@drone:~$ rostopic echo /tello/states
header:
  seq: 1
  stamp:
    secs: 145
    nsecs: 436000000
  frame_id: "map"
pose:
  position:
    x: 0.915806298363
    y: 1.01122386496
    z: 0.0995096131857
  orientation:
    x: 0.000366317560029
    y: 0.000238645090948
    z: 0.709997508496
    w: 0.704204051955
...
header:
  seq: 2
  stamp:
    secs: 145
    nsecs: 440000000
  frame_id: "map"
```

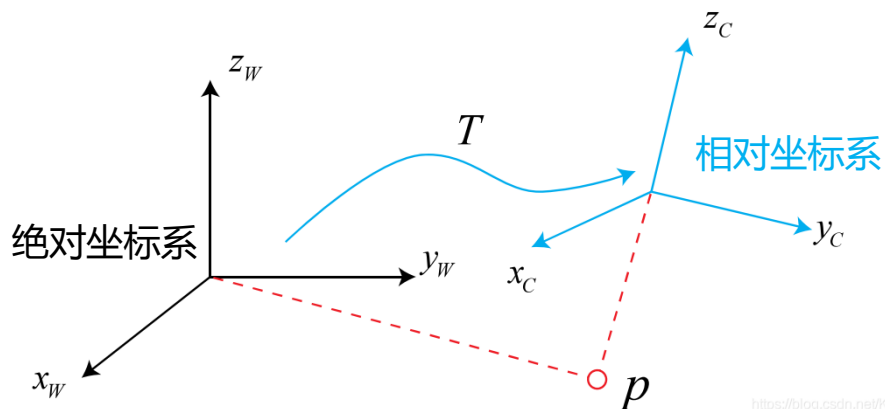
话题/tello/states的实时消息



2 插曲—欧拉角与四元数

• 坐标空间欧式变换

- 欧式变换可以分解为：一次平移 + 一次旋转
- 欧拉角提供了一种非常直观的方式来描述旋转：
 - 绕物体的 Z 轴旋转，得到偏航角yaw；
 - 绕旋转之后的 Y 轴旋转，得到俯仰角pitch；
 - 绕旋转之后的 X 轴旋转，得到滚转角roll。
- 四元数同样可以描述旋转，并且不具有奇异性（欧拉角有奇异性）
- 四元数与旋转矩阵、旋转向量均可相互转换



```
5 from scipy.spatial.transform import Rotation as R
6
7 rotate_1 = R.from_quat([0, 0, 0, 1])
8 rotate_2 = R.from_euler('zyx', [0, 0, 0], degrees=True)
9
10 (yaw, pitch, roll) = rotate_1.as_euler('zyx', degrees=True)
11 (x, y, z, w) = rotate_2.as_quat()
```



2 数据采集——位姿信息

• 数据采集——位姿信息

- 以python为例，读取无人机位姿信息的方式为：
 - 首先订阅话题/tello/states;
 - 有消息到来时，取出其位姿信息;
 - 从位姿信息中读出位置和姿态角信息进行处理。

```
1  #!/usr/bin/python
2  #-*- encoding: utf8 -*-
3
4  import rospy
5  import numpy as np
6  from gazebo_msgs.msg import ModelStates
7  from scipy.spatial.transform import Rotation as R
8
9  def gazeboposeCallback(msg):
10     #取出无人机位姿信息
11     pose = msg.pose
12     position = np.array([pose.position.x, pose.position.y, pose.position.z])
13     #将四元数转化为scipy中的对象，便于后续处理
14     orientation = R.from_quat([pose.orientation.x, pose.orientation.y, pose.orientation.z, pose.orientation.w])
15     print("position ", position)
16     print("orientation ", orientation)
17
18     rospy.init_node('get_pose_node', anonymous=True)
19     gazeboposeSub_ = rospy.Subscriber('/tello/states', ModelStates, gazeboposeCallback)
20     rospy.spin()
21
```




2 数据采集——图像信息

• 数据采集——图像信息

- ROS中的图片消息一般并不是单纯的图片，还包括了一些信息头、图像尺寸等信息，因此（以python为例）需要引入包 `cv_bridge` 进行格式转换；
- 示例见下一页；

```
lee@drone:~$ rostopic info /iris/usb_cam/image_raw
Type: sensor_msgs/Image

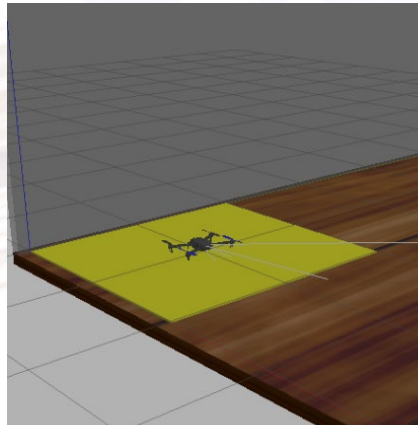
Publishers:
 * /gazebo (http://drone:39389/)

Subscribers:
 * /rviz (http://drone:41567/)

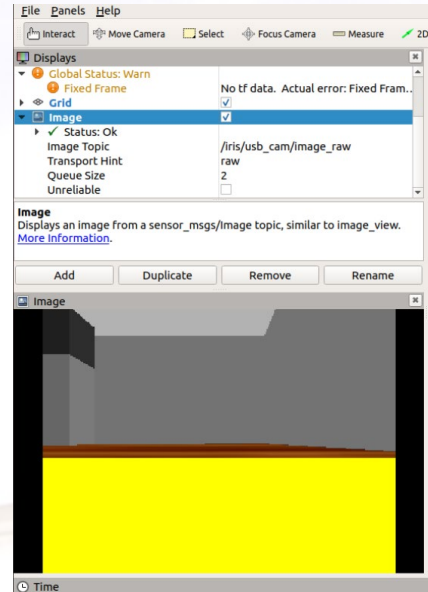
lee@drone:~$ rosmmsg show sensor_msgs/Image
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data

lee@drone:~$
```

话题/iris/usb_cam/image_raw的消息格式



Gazebo第三人称视角



Rviz显示无人机拍摄图像

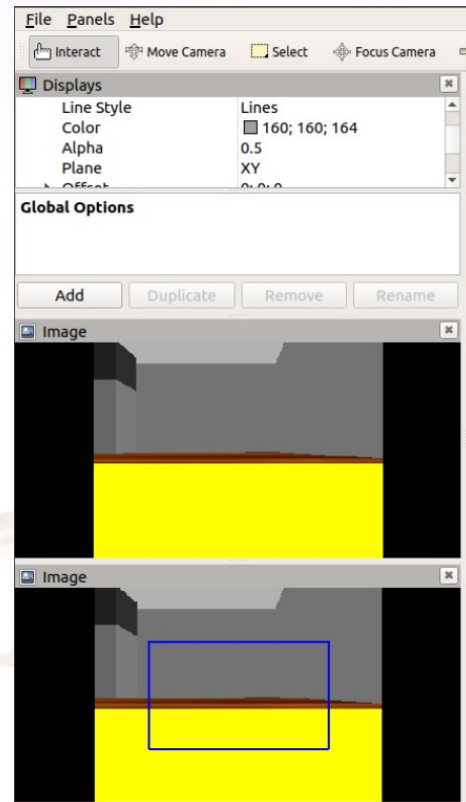


2 数据采集——图像信息

• 数据采集——图像信息

- 以python为例，读取无人机拍摄图片的方式为：
 - 首先订阅话题/iris/usb_cam/image_raw；
 - 有消息到来时，首先用 CvBridge 将消息转换为BGR格式的图片；
 - 对图片进行处理后再发布到指定话题上。
- 下面的例子中在原始图像中间画了一个蓝色矩形框，rviz中效果如右图：

```
1  #!/usr/bin/python
2  #-*- encoding: utf8 -*-
3
4  import rospy
5  import cv2
6  from sensor_msgs.msg import Image
7  from cv_bridge import CvBridge, CvBridgeError
8
9  def imagesubCallback(data):
10     try:
11         bridge_ = CvBridge()
12         #将sensor_msgs/Image类型的消息转化为BGR格式图像
13         orgFrame_ = bridge_.imgmsg_to_cv2(data, 'bgr8')
14         #打印图像尺寸
15         orgFrame_copy = orgFrame_.copy()
16         print("size of raw image: ",orgFrame_copy.shape)
17         #在原始图像上画上矩形框
18         cv2.rectangle(orgFrame_copy, (100, 100), (500, 300), (255,0,0), 2)
19         #将BGR图像再转换为sensor_msgs/Image消息格式并发布
20         image_result_pub_.publish(bridge_.cv2_to_imgmsg(orgFrame_copy))
21     except CvBridgeError as err:
22         print(err)
23
24
25     rospy.init_node('get_image_node', anonymous=True)
26     image_sub_ = rospy.Subscriber("/iris/usb_cam/image_raw", Image, imagesubCallback)
27     image_result_pub_ = rospy.Publisher("/get_image/image_processed", Image, queue_size=10)
28     rospy.spin()
29
```





目 录

- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结



2 飞行控制

• 飞行控制

- 仿真环境中封装了常用的tello命令
- 向话题 /tello/cmd_string 发布对应的字符串消息即可(控制频率**不超过2Hz**)

接口	含义
takeoff	起飞
land	降落
up x ($20 \leq x \leq 500$, 单位cm)	上升 x cm
down x	下降 x cm
forward x	向前 x cm
back x	向后 x cm
left x	向左 x cm
right x	向右 x cm
cw x ($1 \leq x \leq 360$, 单位deg)	顺时针旋转 x deg
ccw x	逆时针旋转 x deg
stop	在当前位置悬停



2 飞行控制

• 飞行控制（命令行演示）

- 仿真环境中封装了常用的tello命令
- 向话题 /tello/cmd_string 发布对应的字符串消息即可(控制频率不超过2Hz)
- 可以通过命令行直接向无人机发布控制指令

```
rostopic pub -1 /tello/cmd std_msgs/String "takeoff"
```

- 仿真环境演示



2 飞行控制

• 飞行控制——路径规划

- 可以预先为无人机指定路径，飞行过程中只需要逐步到达各个目标节点即可，也可以在飞行过程中动态规划飞行路径，这将涉及到路径规划算法。
- 若预先规划路径，只需要将关键的节点坐标保存在 /uav_sim/config/config.yaml 文件中，在launch中载入该文件，即可读取为ROS参数：

```
<node pkg="uav_sim" type="core_controller.py" name="core_controller" output="screen">  
  <rosparam command="load" file="$(find uav_sim)/config/config.yaml" />  
</node>
```

- 在python程序中可以通过以下代码获取该参数。

```
rospy.get_param('/core_controller/trajectory')
```

```
trajectory: [[1.,1.,2.,0.],  
             [1., 4.5, 1., 0.],  
             [1., 4.5, 1., 90.],  
             [1., 4.5, 1., 180.],  
             [1., 4.5, 1., 270.],  
             [1., 4.5, 1., 0.],  
             [1., 9.5, 1., 0.]]
```



目 录

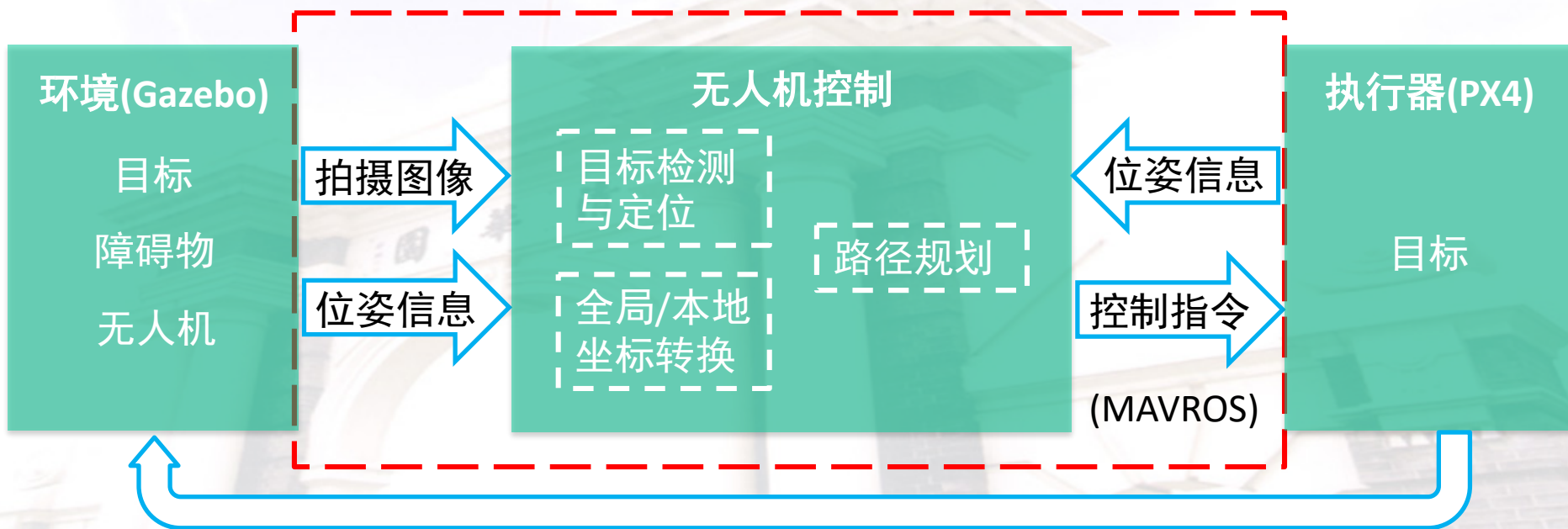
- 1 认识仿真界面
- 2 基本功能介绍
 - 2.1 控制流程
 - 2.2 数据采集
 - 2.3 飞行控制
- 3 小结



3 小结

• 飞行控制流程

- 利用前面的基本操作，集成完整的无人机控制流程，下图虚线矩形框中就是我们要做的主要工作：
 - 收集自身与周围环境的目标信息
 - 根据收集信息决定控制指令
 - 发布控制指令并执行





实验要求

• 课上实验要求

- 利用前面的基本操作，集成完整的无人机控制流程：
 1. 结合tello的控制接口，控制无人机从指定位置起飞；
 2. 识别模拟火情标记（红色）；
 3. 穿过其下方对应的窗户，并在指定位置降落。
- Demo
 - 首先补全 scripts/controller.py 文件
 - 命令终端启动launch文件

```
roslaunch uav_sim windows.launch
```

- 启动新的终端发布开始命令

```
rostopic pub /tello/cmd_start std_msgs/Bool "data: 1"
```

- 启动controller节点，完成飞行任务。



谢谢!



附录：飞行控制

• 飞行控制——原理

- 无人机的飞行控制由PX4自动驾驶仪固件完成，PX4有多种不同的**飞行模式**（官方文档https://docs.px4.io/v1.9.0/en/flight_modes/index.html）
 - **手动模式(Manual)**：通过手动的输入直接产生期望速度（比如摇杆）；
 - **外部控制(Offboard)**：外部基于**MAVLink通信协议**，输入期望的位置、速度或者姿态信息（一般由计算机程序给出）；
 - **自动模式(Auto)**：自动模式又包含多种不同子模式，比如起飞、降落等；
 - Others ...
- 我们在仿真实验中多采用**外部控制模式**，习惯上将控制指令发送方称为**地面站**，因此就需要地面站与无人机之间通过MAVLink通信协议进行通信。
- MAVROS则是MAVLink协议在ROS中的封装，因此要想控制无人机，就需要借助MAVROS向无人机发送控制指令，然后由PX4执行。

