**初赛仿真环境使用说明**

**一、配置仿真环境**

首先，确保你已完成了仿真环境的基本配置（课程组提供的配好环境的虚拟机已经配置完毕，无需其他操作。使用自己环境的同学参考“环境配置方法.doc”）。网盘上已经更新了可用于初赛仿真的ROS工作空间catkin\_sim，包含先前的仿真例程、初赛场景实例、裁判机等。

运行以下命令：

roslaunch uav\_sim arena\_with\_car.launch

即可启动一个初赛赛场的仿真界面。

**二、初赛仿真环境介绍**

**2.1 初赛基本流程**

初赛的基本流程如下：比赛开始前，工作人员将随机生成一个仿真场地文件；每次随机生成过程中，场地中的障碍物位置不变，但会随机在5个给定位置中随机选取3个生成红、黄、蓝三种颜色的球作为待检测目标，以及在楼外墙体上三个窗户上随机生成一处标记作为模拟着火点，门口上方会生成一出模拟着火点；无人车和无人机在接收到裁判机发出的起始信号后能够自行出发，识别模拟着火点，无人机通过对应窗户，无人车通过楼门进入楼内区域；然后，无人车和无人机在楼内区域巡航，判断5个给定位置处是否存在待识别目标以及目标具体类型，并最后在各自指定区域安全停靠落地，比赛结束。

**2.2 赛场基本信息**

赛场示意图如图1所示。整个赛场为一个的矩形（单位为米，下同）。无人机从图1中左上角的绿色区域出发，终点为图1中右下角的紫色区域。无人机的初始坐标为，初始偏航角为，对应朝向轴正方向。无人车初始坐标未(4,1,0)，初始面朝y轴正方向。无人机终点坐标为(7,14.5,0)，无人车终点坐标为(4,14.5,0)。

整个赛场由至处的墙体分为楼内、楼外两个区域。墙体上共有三扇窗户，分别位于至、至、至的区域内，高度范围均为至。三扇窗户中的一扇的正上方会生成一个红色圆形标记，作为模拟着火点。该标记位于对应窗户中心点的正上方处，半径为。墙体上还有一扇门，位于x=3.5至x=4.9的区域内，高度0.25。门的正上方会有一个红色圆形标记，作为模拟着火点。

进入楼内区域后，无人车和无人机需要对5个位置处可能存在的目标进行搜索与识别。其中3个位置会放有分别为半径的红、黄、蓝颜色的小球，另外2个位置不放置目标。目标可能出现的位置如表1所示。

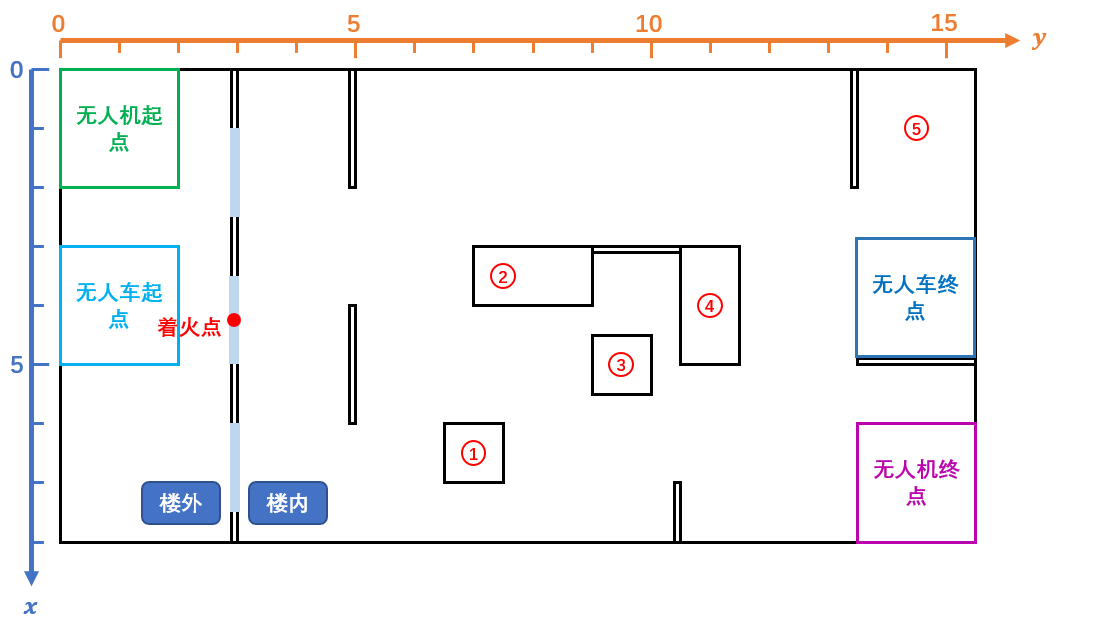


图1 比赛场地示意图

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位置编号 |  |  |  | 位置描述 |
| 1 |  |  |  | 位于一个高处的灰色单层柜子中 |
| 2 |  |  |  | 位于一个木制四层柜子中（一层或者四层） |
| 3 |  |  |  | 位于一个咖啡桌上 |
| 4 |  |  |  | 位于一个木制四层柜子（一层或者四层） |
| 5 |  |  |  | 位于墙角处 |

表1 目标可能出现的位置

ROS包uav\_sim提供了两个初赛赛场环境的实例，分别为car\_arena1.world和car\_arena2.world，存放在uav\_sim/world中。以car\_arena1.world为例，可以通过如下命令在Gazebo图形界面中查看仿真环境：

roscd uav\_sim

gazebo world/car\_arena1.world

**三、仿真环境的使用**

**3.1 裁判机**

更新后的uav\_sim包中提供了初赛使用的裁判机。裁判机共有以下三个功能：

·发出比赛开始信号，并开始比赛计时；  
 ·检测无人机降落和无人车停靠，并停止计时，计算比赛所用时间；  
 ·接收发布的目标检测结果（相关格式及topic见附录A）。

以课程中提供的简单无人机仿真例程（windows.launch）为例。可以在

roslaunch uav\_sim windows.launch

后，打开一个新终端，运行命令：

rosrun uav\_sim judge.py

即可打开一个裁判机窗口。可以按照该窗口的指示，尝试使用裁判机。

打开裁判机后会要求按任意键，按键后裁判机向/tello/cmd\_start发布消息开始任务。只有在裁判机发信号开始后无人车和无人机才可以移动。无人车巡视完毕准备驶向终点时，需要在/AKM\_1/parkstate中发布“PARKING”消息，通知裁判机。

目标检测结果的ground truth存储在uav\_sim/config/target.yaml中。更换赛场环境时需要修改该文件。

**3.2 修改launch文件**

比赛时，需要roslaunch同学们自己写的launch文件，一次性地把仿真所需的所有ROS节点运行起来。可以在uav\_sim中的windows.launch或arena\_with\_car.launch的基础上进行修改。

一方面，需要告诉ROS我们要使用的是哪个赛场环境。windows.launch中有如下语句：

<arg name="world" default="$(find uav\_sim)/world/windows.world"/>

该语句声明了一个参数world，默认值为uav\_sim/world/windows.world。如果需要使用其他赛场环境，可以直接在launch文件中修改该参数的默认值，或是在roslaunch时指定参数值，如：（以下为一行命令）

roslaunch uav\_sim windows.launch world:=/home/thudrone/catkin\_sim/src/uav\_sim/world/car\_arena1.world

注意这种方法需要指定赛场环境文件的绝对路径。

另一方面，需要在launch文件中加入控制无人机需要用到的一个或多个节点。windows.launch的控制节点只有一个controller.py，文件中的相关语句如下：

<node pkg="uav\_sim" type="controller.py" name="controller" output="screen"> </node>

可以参考此格式自行增删节点。

**附录A：仿真环境中可能用到的ROS topic**

**A.1 由仿真环境发布**

A.1.1 /iris/usb\_cam/image\_raw

类型为sensor\_msgs/Image。包含无人机前置摄像头的图像信息。

A.1.2 /tello/states

类型为geometry\_msgs/PoseStamped。包含了无人机当前6D位姿信息。

A.1.3 /tello/cmd\_start

类型为std\_msgs/Bool。由裁判机发布的开始比赛命令。

A.1.4 /AKM\_1/camera /image\_raw

类型为sensor\_msgs/Image。包含无人车前置摄像头的图像信息。

A.1.5 /AKM\_1/odom

类型为nav\_msgs/Odometry。包含了无人车当前的位置信息。

**A.2 由仿真环境订阅**

A.2.1 /tello/cmd\_string

类型为std\_msgs/String。无人机可向此topic发布tello格式控制命令，以实现对无人机飞行的控制。

A.2.2 /target\_result

类型为std\_msgs/String。无人机或无人机向此topic发布目标识别、检测的结果。发布信息应为长度为5的字符串，字符串的每一位代表对应位置的目标检测结果：红球为‘r’，黄球为‘y’，蓝球为‘b’，没有目标为‘e’。例如，检测到位置1为红球，位置2为黄球，位置3为蓝球，则正确结果应为“rybee”。

A.2.2 /AKM\_1/cmd\_vel

类型为geometry\_msgs/Twist。向此topic发布无人车的速度控制信号。

A.2.2 /AKM\_1/parkstate

类型为std\_msgs/String。无人车准备向终点停靠时向此topic发布“PARKING”，以通知裁判机无人车马上完成任务。