公开

2024年（第二届）“洪都杯”智能空战大赛
参赛选手手册

单位：航空工业洪都

时间：2024年5月

目　录

1. 比赛场景介绍

1.1 场景介绍

1.1.1 作战区域

竞赛采用红蓝双方 6V6 对抗模式，红蓝双方初始距离 100km（交战区域为 100*100km 的矩形，中间有半径为 15km 的圆柱形交战热区，俯视图见图 1），战场中心设在（115.86° E，28.68° N）的位置。

1.1.2 兵力设置

双方初始区域为直径 5km 的半圆形区域，初始高度在 8000m-12000m 内随机，速度在 0.8Ma-1.2Ma 随机，双方对头且双方初始态势相同（例：红方有人机初始高度为 8000m，则蓝方有人机初始高度也为 8000m），双方各操控 2 架有人机与 4 架无人机（有人机与无人机之间除武器载荷不同外，其余特性均相同），其中有人机携带 2 枚中距弹和 2 枚近距弹，配装火控雷达、光电传感器和雷达告警系统；无人机携带 4 枚近距弹，配装火控雷达、光电传感器和雷达告警系统，红蓝双方均可获得预警机给出的态势信息（无法用于火控引导）。

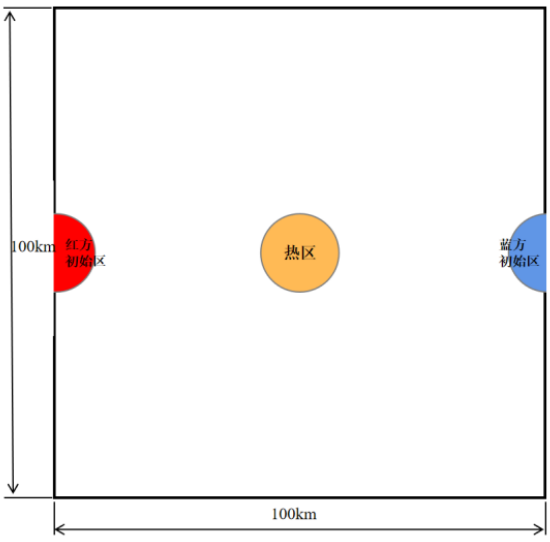

图 1 作战想定俯视图

1.1.3 比赛评分细则

1.1.3.1 基本规则

竞赛中，单批仿真对抗最大时间限制为 10 分钟。当一方飞机被空空导弹命中、飞机飞行至仿真限制区域外（限高 15000m）都会被判定为出局。

1.1.3.2 空空导弹攻击规则

空战竞赛单批仿真对抗中，当火控系统解算满足导弹攻击条件，可发射空空导弹对敌实施攻击。参赛双方飞机被敌方空空导弹命中则判定为击落。

1.1.3.3 热区抢位规则

竞赛对抗中，除使用机载武器对敌方实施攻击外，还将统计参赛双方的热区抢位评分，这适用于双方战损相同时判定双方胜负。判定双方热区抢位评分计算方法如下：

$$E = \frac{T_a}{T}$$

其中，T_a 为一方在仿真过程中在热区的占位时间，T 为仿真总时间。

1.1.3.4 仿真结束条件

基于上述规则，在单批对抗仿真中，出现下列情况之一将终止当前批次仿真：

终止条件 1：至少一方所有飞机出局（含被击落、超出仿真限制区域），且空中没有正在飞行且未脱靶的空空导弹时；

终止条件 2：当前批次仿真超最大时间限制，且没有正在飞行的空空导弹时。

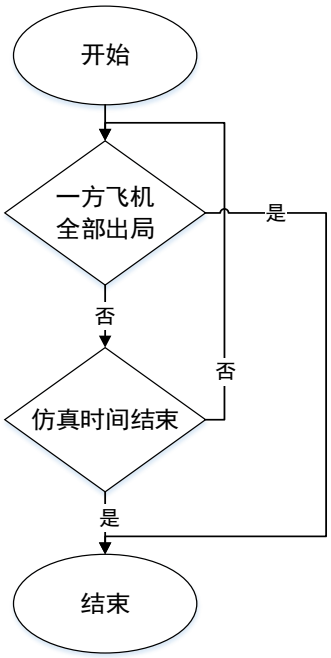

图 2 终止条件判定

1.1.3.5 单批仿真胜负判定条件

仿真终止后，通过以下方式判断单批次仿真胜负关系：

（1）若双方有人机存活数量不同，则场上存在有人机的一方为胜方；

（2）若双方有人机存活数量相同，则场上无人机存活数量高的一方为胜方；

（3）若双方有人机、无人机存活数量均相同，则热区争夺时间长的一方为胜方。

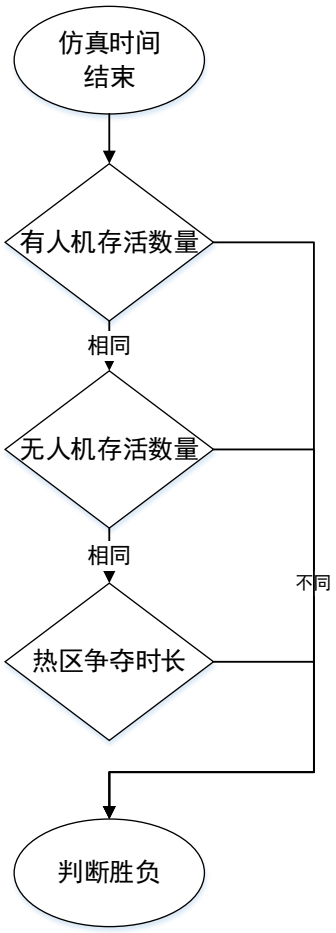

图 3 双方胜负判定

1.1.3.6 总分评定方法

对战双方共完成多批次仿真对抗，初始条件在限制范围内随机。当任意两个参赛队伍的所有仿真对抗批次结束后，统计双方获胜批次数量，获胜批次数多者为本场对抗赛胜方。胜方获得 2 分积分，平局得 1 分，负方得 0 分。

当组内所有参赛队伍完成两两对抗后，依据各队所获积分总数给出竞赛排行榜名次。当两队积分相同时，按照获胜批次数多者排名靠前确定各队名次，若双方获胜批次相同，则依次按照双方所有对局中击落有人机总数、击落飞机总数、占领热区总时间排序。

1.2 仿真系统介绍

本平台是一款适用于多个场景的智能空战仿真系统，集成了以飞机、空空导

弹、雷达为主的多种模型，极大程度地还原了真实场景下的战况。除此之外，该仿真系统还提供了可视化模块，为选手探索、分析以及优化智能体提供了强有力的支持。

1.2.1 仿真说明

（1）时间：态势中给出的时间均为“仿真时间”，以场景开始时为 0 时刻，以秒为单位给出相对于 0 时刻所经过的时间。

（2）单位：态势主要采用公制单位包括米、秒等，角单位多采样弧度制。经度、纬度等信息采用角度制。

（3）坐标系：仿真中给出的坐标（x-y-z）为北东地坐标。其坐标原点在东经 115.86 度，北纬 28.68 度，高度 10000 米处。

（4）信息精度：对抗中使用的态势信息由于获取途径的不同，存在三种精度概念：

1）精确完全信息：可获取包含位置、姿态以及角速度、锁定列表挂载状态等全部属性信息。

2）精确不完全信息：仅可获取精确无误差的位置信息。（由于传感器能力或其它原因无法知晓敌机的其他属性信息，例如敌方的角速度、挂载、锁定状态等）

3）粗略信息：仅可获取有一定误差的位置信息。（信息来源于告警和预警系统）

1.2.2 飞机平台

飞机平台采用全量六自由度仿真模型，可接受杆、舵级控制指令输入。其中选手可以通过编写程序向仿真模型发送对应的指令参数，参数包括：滚转、俯仰、偏航、油门杆位移（详见 3.3 指令接口说明）。对抗飞机仿真模型的部分属性见表 1。

表 1 飞机模型的部分属性

属性名	单位	属性值
机翼面积	平方米（m^2）	27.87
翼展	米（m）	9.14
机翼弦长	米（m）	3.45

水平尾翼面积	平方米（m^2）	5.92
水平尾翼力臂	米（m）	5.02
垂直尾翼面积	平方米（m^2）	5.09
军用推力	磅力（lbf）	17800.0
最大推力	磅力（lbf）	29000.0

1.2.3 雷达

（1）雷达特性：仿真中每架飞机均搭载一个火控雷达。雷达的探测范围为一个 120 度角的扇面，其俯仰范围为[-10, 32]度。雷达仅能探测飞机目标，无法探测到导弹目标。

（2）RCS 特性：飞机前向±30 度范围内 RCS 均值为 0.1 平方米量级。飞机的 RCS 由表格插值计算，表格的两个查找键值是照射飞机在被照射飞机的机体轴坐标系下的 z 轴旋转角度和 y 轴旋转角度。

（3）雷达与 RCS 交互逻辑。通过雷达波发射能量和对方飞机的相对位置姿态解算出最大发现距离 R_{max}，计算方式如下：

$$R_{max} = K * \sqrt[4]{RCS}$$

其中，K 为常数 106066，R_{max} 的单位为米。如果对方飞机距雷达载机的距离小于 R_{max}，且位于载机雷达的探测扇面内，则认为雷达探测到对方飞机。仿真平台将返回目标信息。该目标信息为精确不完全信息，可用于中距弹的中制导过程。

1.2.4 光电传感器

（1）仿真中每架飞机均搭载一个光电传感器。光电传感器的探测范围为 7.5 度，距离为 10km 的固定圆锥。传感器仅能探测飞机目标，无法探测导弹目标。

（2）仿真中，目标飞机落在载机光电传感探测锥内，即认为目标飞机被光电传感器探测到，并返回目标信息。该目标信息为精确不完全信息。

仿真平台中，由雷达和光电传感器探测到的信息不会作为单独的结构体输出，其信息经过态势整合可以从我方飞机态势信息中获取（详见 3.2 态势信息接口说明）。

1.2.5 空空导弹

（1）中距空空导弹：

仿真平台采用通用雷达导引主动中距空空导弹仿真模型，由载机火控雷达提供目标信息用于火控解算及中制导。

导弹发射条件为：载机火控雷达探测到目标，且该目标在载机雷达探测扇面内连续时间超过 0.4s，且该目标与载机的距离小于导弹最大发射距离。

导弹是否脱靶判断条件：中距空空导弹进入中制导后，若己方雷达探测不到目标持续时间超过 1.0 秒，中制导失效脱靶，并且导弹未来无法再进入制导状态。

导弹进入末制导条件：导弹距离目标飞机小于 20km。导弹进入末制导后，不再需要载机雷达提供目标信息。

在仿真平台中，经火控解算后可向用户直接提供武器是否可发射、满足武器发射条件的目标数量以及各目标编号。导弹发射后，在导弹的状态信息中可查看导弹是否进入末制导、是否已经脱靶等信息。且导弹仅能通过载机上的火控雷达获取制导信息。

（2）近距空空导弹：

采用通用红外近距空空导弹仿真模型，由载机光电传感器提供位置信息用于火控解算。

导弹发射条件为：载机光电传感器探测到目标，且该目标在载机光电传感器探测锥内被连续探测时间超过 0.25s。

导弹发射后无需载机控制，由导弹的红外导引头自主导引。

（3）空空导弹发射间隔：每架飞机在发射导弹后有 0.5 秒的发射间隔，此时间内不能再发射导弹。

（4）命中毁伤：空空导弹与目标在一定范围内，系统将判定导弹命中，目标飞机被摧毁。被摧毁目标在双方态势中均会彻底消失。

1.2.5 雷达告警系统（RWS）

受到敌方雷达照射时，己方能够获取由雷达告警系统返回的敌方照射单位（进入末制导的中距空空弹、敌方飞机）的**粗略信息**。雷达告警系统是我方获取敌方中距导弹信息的唯一手段。敌方中距空空导弹在中制导过程中我方无法获取敌方导弹信息。

为方便选手训练，雷达告警系统也将给出敌方近距空空导弹的粗略信息。

1.2.6 预警系统（AWACS）

红蓝双方均有场外预警机提供预警信息，该信息能为我方提供敌方飞机（不包括空空导弹）的**粗略信息**。该信息的推送频率为 0.1 帧/秒（即 10 秒 1 次）。

1.3 选手使用系统总体框架

仿真对战环境的总体架构如图 4 所示，其中蓝色的部分为主办方提供的模块，黄色的部分为选手设计实现的模块，绿色的部分表示外部可视化软件。主办方将为选手们提供一个测试目录，其中包括仿真环境 HDDF2Sim、简单智能体示例和测试脚本。在开发阶段，选手需要将仿真环境拷贝到自己的开发目录下，编写仿真交互接口，设计开发智能体模型，并自定义训练框架提升优化智能体。测试阶段，选手将训练好的智能体及其辅助模块拷贝到测试目录下进行测试，并通过外部实时态势显示软件 Tacview 观看对战信息。

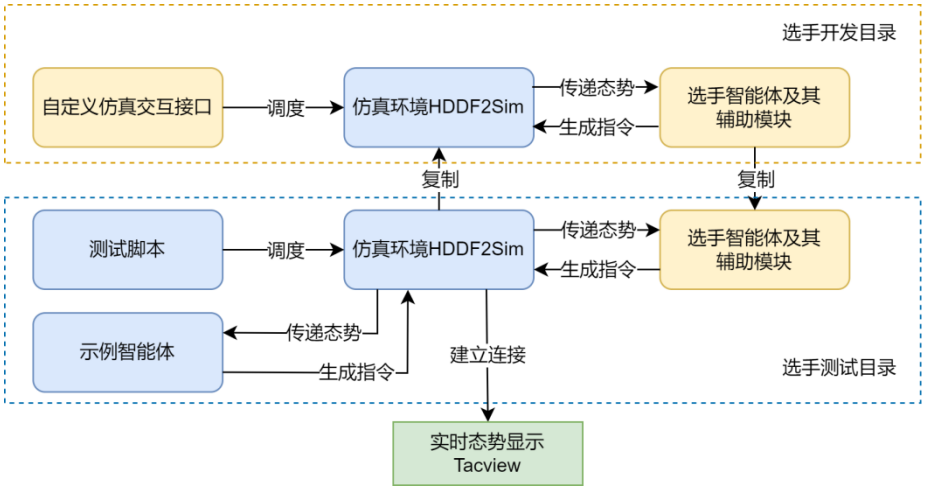

图 4 平台总体架构图

1.4 可视化模块介绍

基于 Tacview 软件仿真平台提供了作战实时遥测与作战数据回放两种可视化方式，下面简要介绍在 Tacview 中作战仿真常用的部分功能。

1.4.1 雷达

仿真中飞机针对全方向的雷达反射特性进行了建模。Tacview 可视化中，红方飞机未被蓝方雷达发现时机体呈现红色；红方飞机被蓝方雷达发现时其机体呈亮粉色；蓝方飞机未被红方发现时机体呈现蓝色；蓝方飞机被红方雷达发现时机体呈现亮绿色。

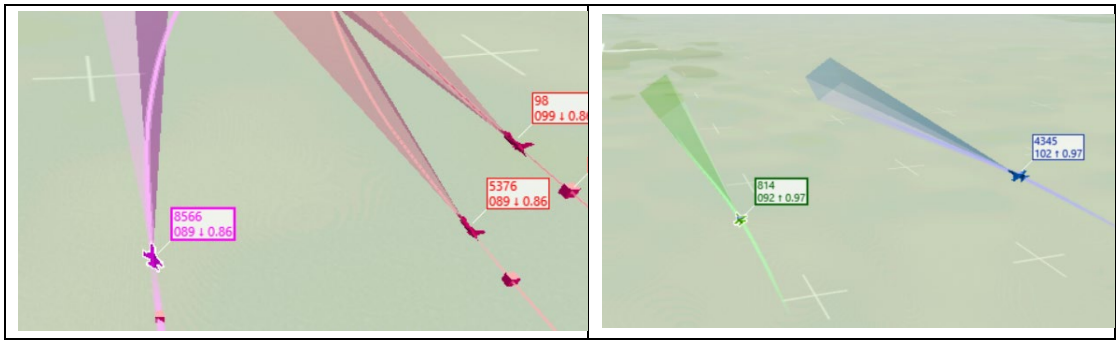

图 5 雷达可视化效果图

1.4.2 中距空空导弹

中距空空导弹需要飞机雷达提供中制导，中制导持续的条件为敌机处于“被发现”状态的间断时间不能超过 1 秒，否则脱靶。中距空空导弹进入距离敌机 20km 后进入末制导，此时不再需要飞机雷达导引。

图 6 中距弹可视化效果图

1.4.3 近距空空导弹

近距空空导弹的发射条件较为苛刻，发射条件为在可视化中敌方处于飞机前方示意锥体中且持续被探测约 0.25 秒。中距锁定和近距锁定状态在态势中会直接给出态势整合后的结果。选手不需要关心制导锁定的具体过程。

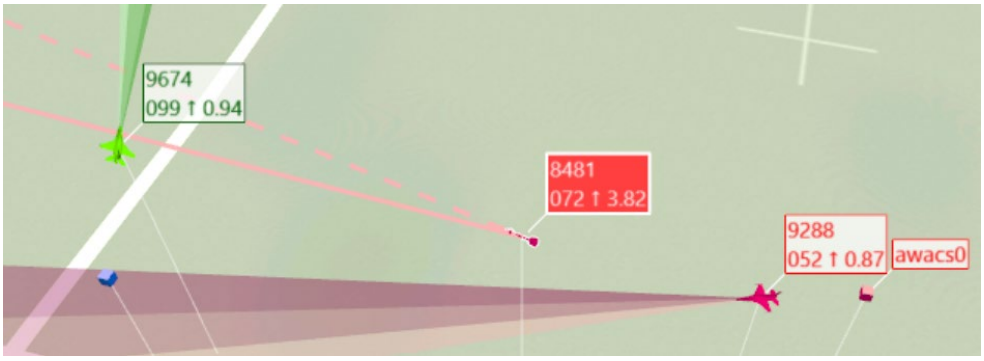

图 7 近距弹可视化效果图

1.4.4 预警信息

仿真给态势推送一个 0.1 帧/秒的预警机获取的敌方粗略位置信息。该信息在

可视化中以小方块的形式标注。

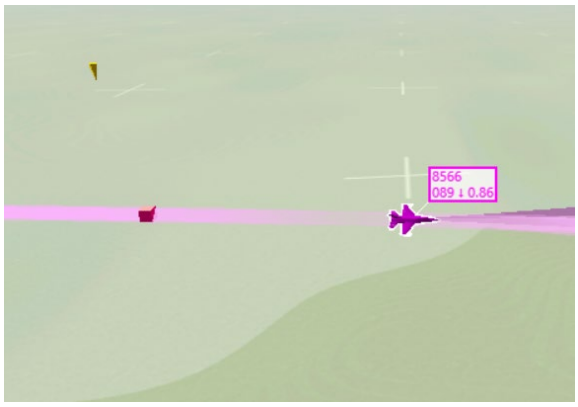

图 8 预警信息可视化效果图

1.4.5 事件日志

可视化中会给出本局推演的事件日志，可以通过工具栏中的“分析”->“事件日志”将其调出。事件日志会打印包括空空导弹发射、毁伤、对局结束评分及评分依据等信息。

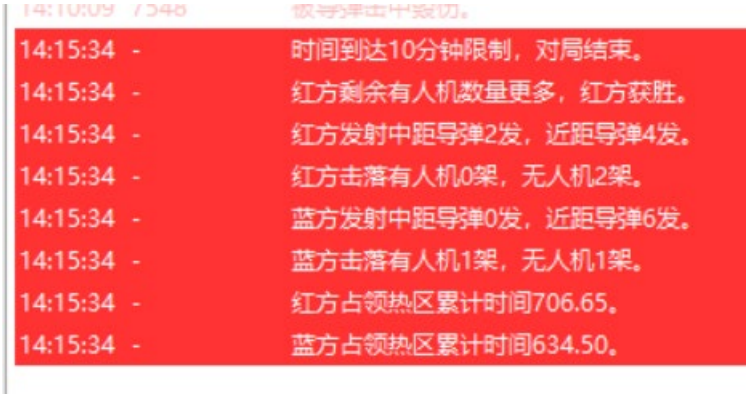

图 9 事件日志可视化效果图

1.4.6 其他辅助功能

在可视化软件 Tacview 中，还有许多辅助功能可供选手们使用。这里列举比较常用的几种，如图 10 所示。

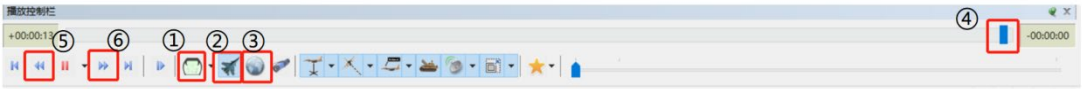

图 10 可视化软件中的常用功能

选手们可以通过改变座舱视角来观察飞机的姿态，图 11 中的①，②和③分别为座舱视野、外部视野和卫星视野，如图 11 所示。

（a）座舱视野　　（b）外部视野　　（c）卫星视野

图 11 三种不同视角效果图

除此之外，选手可以通过拖动图 10 中的④进行回放展示，⑤和⑥可以减慢或加快对战演示。当选手想要对某一架飞机或某一枚空空导弹进行观察评估时，可以自行选择聚焦的目标，如图 12 所示。

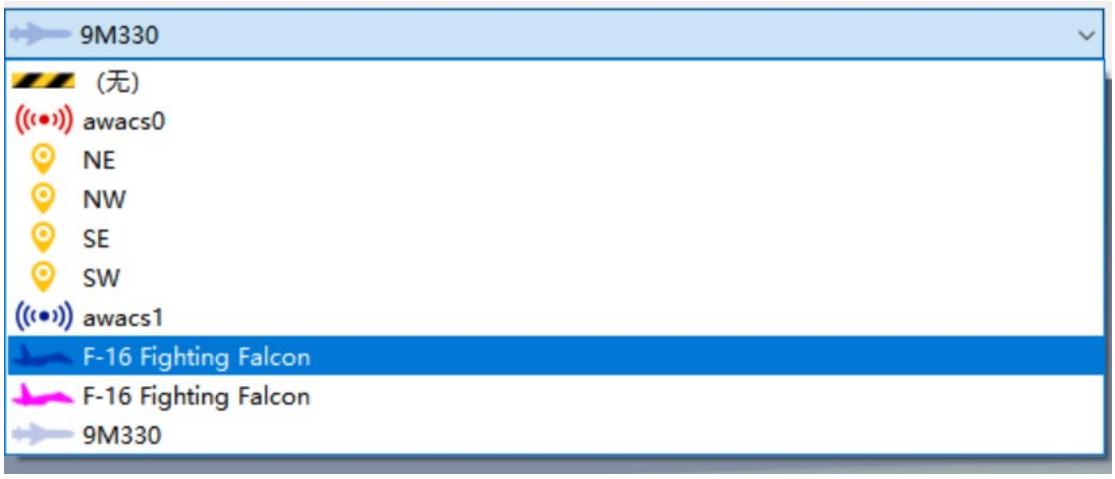

图 12 选择聚焦的目标

2. 运行环境介绍

2.1 运行环境需求

系统环境：Windows10 操作系统；

外部软件：Tacview；

Python 包管理软件：Anaconda 或 Miniconda。

2.2 Python 环境配置

（1）平台仅支持 python 3.7.X 版本，依赖包文件见附录 1（平台附有 requirements.txt 文件）。

注意：此次比赛原则上不允许使用主办方提供的附录外依赖包，如选手需要用到附录以外的依赖包，可联系主办方。

（2）主办方提供由 conda 软件配置的 python 虚拟环境，选手也可自行下载安装 python 环境，最终提交的智能体需在主办方提供的测试环境下成功运行并通过测试，无法通过测试的智能体视为放弃比赛资格。

（3）如选手需使用除 python 外的其他语言开发智能体，请自行封装 python 交互接口，并在提交前对智能体进行测试。

2.3 仿真环境安装

主办方提供的平台环境中的 hd-tournament 文件夹包含完整的仿真环境，选手需拷贝该文件夹到自己的开发目录下进行智能体的开发。

3. AI 接口介绍

3.1 仿真接口

选手需要实现对仿真环境的初始化，红蓝双方智能体的初始化以及仿真环境推演等，主要的方法功能和属性含义描述如表 3 所示：

表 2 仿真接口文件的主要方法功能与属性含义

方法（属性）名	传入参数	返回内容	功能说明
HDDF2Sim()	（1）场景设定 （2）可视化开关（默认：True） （3）录像保存开关（默认：False） （4）录像保存路径（默认：None）	仿真环境	初始化仿真环境,场景设定通过 json 文件加载，可视化开关控制是否连接 Tacview 实时查看对局战况，录像保存开关控制是否保存本局录像，如保存录像，可以从录像保存路径下查看到录像回放
RedAgent()	阵营标识	红方智能体	红方智能体初始化
BlueAgent()	阵营标识	蓝方智能体	蓝方智能体初始化
HDDF2Sim.done	—	—	判断对战环境是否推演结束
HDDF2Sim.get_obs()	阵营标识（默认：″none″）	态势信息	获取某一方智能体的当前态势信息
HDDF2Sim. send_commands()	（1）指令信息 （2）执行指令的阵营（默认：″none″）	—	向仿真环境发送某一方执行的指令信息
Agent.step()	智能体对应阵营的态势信息	智能体生成的指令信息	智能体根据当前态势信息生成指令信息
HDDF2Sim.step()	—	—	对战环境推演一个时间步

3.2 态势接口

仿真给选手的态势信息为 Observation 类，其具体数据结构如下表 4 所示：

表 3 Observation 类

成员变量名	含义	类型	单位	备注
sim_time	当前仿真相对时间	float	秒	仿真开始时为 0
side	所属阵营	string		″red″ 或者 ″blue″

my_planes	己方飞机字典	dict		key 为飞机编号，value 为该飞机对应的 MyPlaneInfo 类
enemy_planes	敌方飞机字典	dict		key 为飞机编号，value 为该飞机对应的 EnemyPlaneInfo 类
missile_infos	我方导弹信息	dict		key 为导弹编号，value 返回对应的 MissileInfo 类
awacs_infos	预警信息	list		列表元素为 AWACSSimInfo 类，仅出现敌方信息
rws_infos	雷达告警信息	list		列表元素为 RWSSimInfo 类，返回我方雷达告警系统输出信息

其中，sim_time 为仿真相对时间，每局 sim_time 由 0 开始。side 为态势接收方所操控的阵营。my_planes 和 enemy_planes 为己方飞机和敌方飞机字典，键为单位编号，元素值为飞机对应的 MyPlaneInfo 类和 EnemyPlaneInfo 类，注意，这里获取到的我方飞机信息和敌方飞机信息存在差异。missile_infos 为我方空空导弹信息字典，其中的元素为 MissileInfo 类。awacs_infos 为预警机给出的 0.1 帧/秒的敌机信息。rws_infos 为雷达告警系统给出的敌方位置信息（包含照射我方的敌方飞机、导引头锁定我方的敌方空空导弹）。

3.2.1 飞机信息（MyPlaneInfo 类和 EnemyPlaneInfo 类）

使用 obs.my_planes 和 obs.enemy_planes 可索引到己方飞机的 MyPlaneInfo 类和敌方飞机的 EnemyPlaneInfo 类，其具体数据结构如下表 5 和表 6 所示：

表 4 MyPlaneInfo **类**

己方成员变量名	含义	类型	单位	备注
ind	飞机编号	int		
side	所属阵营	string		″red″ 或者 ″blue″
is_uav	飞机类型标识	string		True 表示无人机，False 表示有人机
x	x 轴坐标值	float	米	北-东-地坐标系，坐标原点位于经度 115.86°、纬度 28.68°、高度 10000m
y	y 轴坐标值	float	米	
z	z 轴坐标值	float	米	
lon	经度	float	度	
lat	纬度	float	度	
height	高度	float	米	
yaw	偏航角	float	弧度	由北往东顺时针为正

pitch	俯仰角	float	弧度	向上为正
roll	滚转角	float	弧度	机头朝向右手螺旋顺时针为正
v_north	北向速度分量	float	米/秒	
v_east	东向速度分量	float	米/秒	
v_down	地向速度分量	float	米/秒	
sp	速度	float	米/秒	地速
alpha	迎角	float	弧度	
beta	侧滑角	float	弧度	
cas	校准空速	float	米/秒	
tas	真实空速	float	米/秒	
mach	马赫数	float		
acc_nz	法向过载	float	g	
omega_p	绕机体 x 轴角速度	float	弧度/秒	
omega_q	绕机体 y 轴角速度	float	弧度/秒	
omega_r	绕机体 z 轴角速度	float	弧度/秒	
loadout	武器挂载	dict		格式为{武器名:剩余数量}，例如{'mid_missile':1, 'short_missile':2} 表示当前该单位剩余 1 枚中距导弹和 2 枚近距导弹
mid_lock_list	满足中距空空导弹解算条件的敌机列表	list		列表中的元素为敌机编号
short_lock_list	满足近距空空导弹解算条件的敌机列表	list		列表中的元素为敌机编号

表 5 EnemyPlaneInfo **类**

敌方成员变量名	含义	类型	单位	备注
ind	飞机编号	int		
side	所属阵营	string		"red" 或者 "blue"
is_uav	飞机类型标识	string		True 为无人机，False 为有人机
x	x 轴坐标值	float	米	北-东-地坐标系，坐标原点位于经度 115.86°、纬度 28.68°、高度 10000m
y	y 轴坐标值	float	米	
z	z 轴坐标值	float	米	
lon	经度	float	度	

lat	纬度	float	度	
height	高度	float	米	
yaw	偏航角	float	弧度	由北往东顺时针为正
pitch	俯仰角	float	弧度	向上为正
roll	滚转角	float	弧度	机头朝向右手螺旋顺时针为正
v_north	北向速度分量	float	米/秒	
v_east	东向速度分量	float	米/秒	
v_down	地向速度分量	float	米/秒	
sp	速度	float	米/秒	地速
alpha	迎角	float	弧度	
beta	侧滑角	float	弧度	

己方飞机能获取的数据信息包括：单位编号、阵营、类型、x-y-z 坐标值、经纬高值、偏航角、俯仰角、滚转角、北-东-地三轴上的速度分量、速度、迎角、侧滑角、校准空速、真实空速、马赫数、法向过载、绕机体轴的角速度、当前剩余武器挂载等信息。敌方飞机除绕机体的角速度、当前剩余挂载信息、法向过载以及空速信息外，其他信息都可以获取到。

3.2.2 空空导弹信息（MissileInfo 类）

obs.missile_infos 索引到己方空空导弹信息，返回元素为 MissileInfo 类的列表。具体数据结构如下表 7 所示：

表 6 MissileInfo **类**

成员变量名	含义	类型	单位	备注
ind	导弹编号	int		
side	所属阵营	string		"red" 或者 "blue"
type	导弹类型	string		"mid_missile"为中距导弹，"short_missile"为近距导弹
x	x 轴坐标值	float	米	北-东-地坐标系，坐标原点位于经度 115.86°、纬度 28.68°、高度 10000m
y	y 轴坐标值	float	米	
z	z 轴坐标值	float	米	
lat	纬度	float	度	
height	高度	float	米	

sp	速度总量	float	米/秒	
v_north	北向速度分量	float	米/秒	
v_east	东向速度分量	float	米/秒	
v_down	地向速度分量	float	米/秒	
launcher_ind	发射导弹的飞机编号	int		
target_ind	目标飞机的编号	int		
radar_on	是否能被雷达告警系统捕获	bool		True 为导弹能被雷达告警系统捕获
miss_target	是否脱锁	bool		True 表示导弹已经脱锁

3.2.3 预警信息（AWACSSimInfo 类）

obs.awacs_infos 索引到己方获取的预警信息（针对敌方飞机），具体数据结构如下表 8 所示：

表 7 AWACSSimInfo **类**

成员变量名	含义	类型	单位	备注
info_time	时间	float	秒	预警信息给出的仿真时间
ind	编号	int		
side	所属阵营	string		"red" 或者 "blue"
x	x 轴粗略坐标值	float	米	北-东-地坐标系，坐标原点位于经度 115.86°、纬度 28.68°、高度 10000m
y	y 轴粗略坐标值	float	米	
z	z 轴粗略坐标值	float	米	
lon	经度	float	度	
lat	纬度	float	度	
height	高度	float	米	

注意：预警信息为粗略信息，提供非准确的位置信息。预警信息给出频率为 0.1 帧/秒，提供 info_time 属性，表示信息为过去的某仿真时间给出。

3.2.4 雷达告警信息（RWSSimInfo 类）

obs.rws_infos 索引返回元素为 RWSSimInfo 类的列表，其具体数据结构如下

表 9 所示。

表 8 RWSSimInfo 类

成员变量名	含义	类型	单位	备注
ind	编号	int		
side	所属阵营	string		"red" 或者 "blue"
x	x 轴粗略坐标值	float	米	北-东-地坐标系，坐标原点位于经度 115.86°、纬度 28.68°、高度 10000m
y	y 轴粗略坐标值	float	米	
z	z 轴粗略坐标值	float	米	
lon	经度	float	度	
lat	纬度	float	度	
height	高度	float	米	
alarm_ind_list	告警对应的己方飞机编号列表	list		存放产生此告警信息的己方飞机编号

注意：雷达告警信息仅能获得目标单位的粗略位置信息。

3.3 指令接口格式

仿真环境接收字典格式的指令，它包含至多两组键值对，"control"键值对是必须存在的，用来控制飞机的机动操纵，如需发射空空导弹，则添加"weapon"键值对控制飞机发射空空导弹。其中，"control"字段的值为列表格式，它包含控制飞机的滚转、俯仰、偏航和油门的四个指标（[aileron, elevator, rudder, throttle]），用于向仿真环境输入飞机的控制参数。其范围及其物理意义如表 10 所示。

表 9 指令的参数范围及物理意义

参数名称	**范围**	**物理意义**
aileron	[-1,1]	滚转
elevator	[-1,1]	俯仰
rudder	[-1,1]	偏航
throttle	[0,1]	油门杆位移

"weapon"字段是字典格式，它包含两组键值对，分别表示飞机发射空空导弹的类型和该空空导弹打击的目标。一个简单的 cmd_dict 指令示例如下所示：

{

2044: {"control":[0,0,0,1], "weapon":{‘type’:’short_missile’, ‘target’:3768}
}

4. AI 开发与测试

选手需要自行创建开发目录，将主办方提供的测试目录下的仿真环境相关文件夹拷贝到自己的开发目录下。其次，选手可以根据需求自定义仿真交互接口，然后进行智能体的设计和开发，总体框架图遵循图 13。

```
|—选手开发目录
|     |—仿真环境相关文件夹
|     |—demo.py
|     |—agents
|     |     |—team_blue
|     |     |     |—blue_agent.py
|     |     |—team_red
|     |     |     |—red_agent.py
```

图 13 总体框架图

下面将以主办方提供的示例智能体为例，详细说明选手如何进行智能体的开发和测试。

4.1 仿真交互

4.1.1 场景设定

选手可以在 scen.json 文件中找到初始设定的场景参数，并可以根据自身需求在规则允许的范围内进行合理的调整，以探索最优的策略。其中，部分场景参数和其意义如图 14 所示。

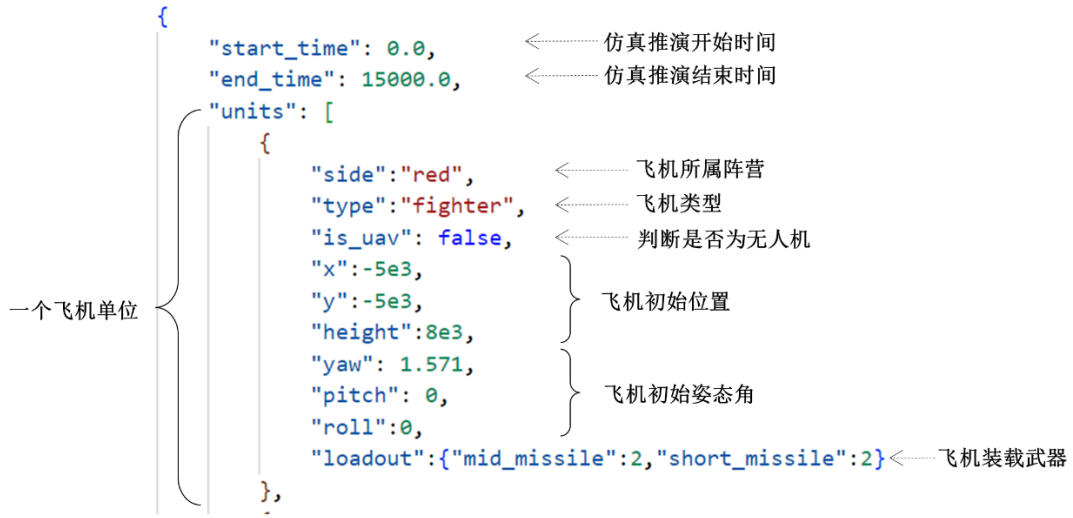

图 14 场景设定的参数

4.1.2 构建仿真交互接口

（1）场景设定完毕后，加载场景文件并保存到变量 scen 中。

```
with open("scen.json", "r") as fin:
    scen = json.load(fin)
```

（2）根据场景参数 scen，可视化标识 use_tacview，录像保存标识和录像保存地址初始化仿真环境。其中，use_tacview 为 True 表示连接可视化软件，save_replay 为 True 表示保存推演的可视化录像。

```
sim = HDDF2Sim(scen, use_tacview=True, save_replay=True, replay_path="replay.acmi")
red_agent = RedAgent('red')
blue_agent = BlueAgent('blue')
```

（3）未达到仿真结束标志前，仿真平台持续推演。红方利用仿真环境的 get_obs()方法获取红方观察的态势，红方智能体根据态势信息，生成红方的指令集合存放到 red_cmd_dict 字典变量中，并通过 send_commands()方法在仿真环境中执行该指令。蓝方智能体根据蓝方返回的态势信息生成相应指令集合并在仿真环境中执行。仿真环境调用 step()方法执行一步推演。

```
while not sim.done:
    cmds = []
    red_obs = sim.get_obs(side='red')
    red_cmd_dict = red_agent.step(red_obs)
    sim.send_commands(red_cmd_dict, cmd_side='red')
    blue_obs = sim.get_obs(side='blue')
    blue_cmd_dict = blue_agent.step(blue_obs)
    cmds.extend(blue_cmd_dict)
    sim.send_commands(blue_cmd_dict, cmd_side='blue')
```

```
    sim.step()
```

4.2 智能体实现

本节以主办方提供的示例智能体为例，向选手展示了结合仿真平台的智能体设计框架，核心是实现智能体类。主办方会提供简单的示例智能体供选手观看效果。注意：大赛要求智能体每一帧运算时间不得超过 50ms。

Agent 类中定义了初始化函数__init__()，函数传入阵营参数，初始化相关变量和工具类、策略状态和统计变量等参数。

```
class Agent:
    def __init__(self, side) -> None:
        self.side = side
        self.rng = np.random.default_rng()
        self.assign = {}
        self.missile_cds = {}
        self.missile_time = {}
```

智能体定义了 step 函数，函数根据传入的态势信息输出智能体的指令。首先按格式创建指令字典变量 cmd_dict，并将当前态势按己方飞机、敌方飞机、预警信息和雷达告警信息进行分类整理。为了方便计算，创建了名为 Vec3 的类，该类支持该智能体中使用到的三维坐标的运算（Vec3 类仅作为示例演示，选手需自行设计实现类似功能）。

```
    def step(self, obs):
        cmd_dict = {}

        allies = obs.my_planes
        for ally in allies.values():
            ally.pos = Vec3(
                ally.x, ally.y, ally.z
            )
        enemies = obs.enemy_planes
        for enemy in enemies.values():
            enemy.pos = Vec3(
                enemy.x, enemy.y, enemy.z
            )
        awacs_infos = obs.awacs_infos
        rws_infos = obs.rws_infos
```

for 循环中对每架己方飞机计算行动。这里以近距空空导弹为例向选手展示了如何控制智能体发射导弹。首先通过监测己方飞机的 short_lock_list 属性列表判断是否满足近距弹的发射条件，如果存在，则该属性列表中会存放满足发射条

件的敌机编号；若该属性列表为空，则不存在任何满足发射条件的敌机。这里还给智能体设置了发弹的冷却时间防止一下打出很多近距空空导弹。除了发射空空导弹的操作，飞机还整合雷达和预警信息，并针对任一敌机做攻击机动。其中attack_move()方法根据当前飞机信息和要追击的敌机信息，计算追击角度，以得到飞机的机动控制参数（此处 attack_move()方法仅作示例演示，选手需自行设计实现类似功能）。

```
        for ally_ind, ally in allies.items():
            weapon_launch_info = {}
            if len(ally.short_lock_list)>0:
                if ally.loadout.get('short_missile', 0)>0 and obs.sim_time -
self.missile_cds.get(ally_ind, 0) > 10.0:
                    weapon_launch_info = {
                        'type': 'short_missile',
                        'target': ally.short_lock_list[0],
                    }
                    self.missile_time[ally_ind] = obs.sim_time

            if len(enemies) or len(awacs_infos):
                if len(enemies):
                    enemy = list(enemies.values())[0]
                else:
                    enemy = awacs_infos[0]
                action = attack_move(ally,  enemy)
            else:
                action = [-1 + 2*self.rng.random(), -1 + 2*self.rng.random(), 0.1 *
np.random.rand(), 0.5 + 0.5 * np.random.rand()]
```

按格式设置每架己方飞机的机动动作和武器指令，指令格式见 3.2 节指令接口格式，返回输出智能体生成的指令集合。

```
            cmd_dict[ally_ind] = {
                'control': action
            }
            if len(weapon_launch_info):
                cmd_dict[ally_ind]['weapon'] = weapon_launch_info
        return cmd_dict
```

4.3 智能体测试

选手将设计好的智能体及其支撑模块代码整合到一个文件夹下，且必须包含__init__.py 文件，在__init__.py 里面导入自己的智能体类名并改名为 Agent，以

保证测试时智能体能够正确被调用，并将该文件夹拷贝到“测试目录根目录/agents/”下。

打开 Tacview 软件，找到“记录->实时遥测”，填写任意用户名，将数据记录地址设置为 localhost 端口号设置为 5555，点击连接。如下图 15 所示：

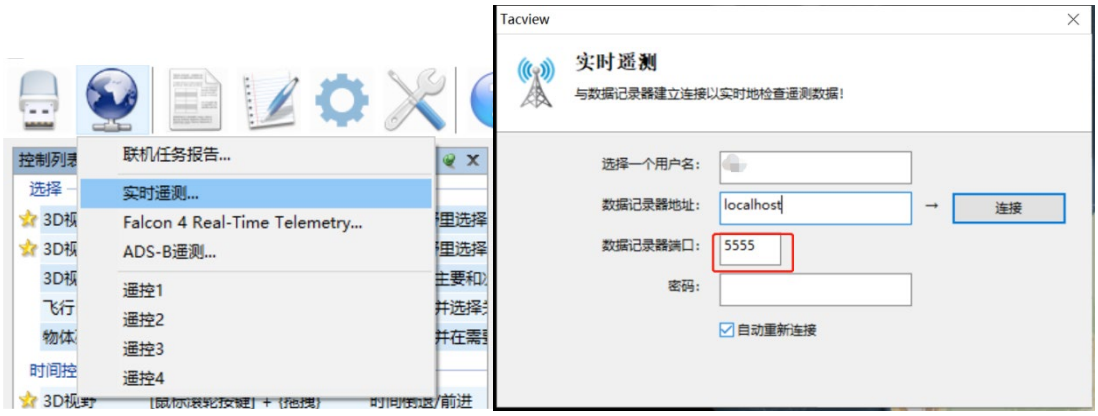

图 15 Tacview 实时遥测

运行测试脚本，运行成功则选手可通过 Tacview 可视化观战对局过程。

4.4 作品提交要求

选手最终的参赛智能体将以文件夹形式提交，并将该文件夹命名为英文战队名称（不能包含特殊字符）。文件夹中的内容要求见 4.3 节智能体测试。

如果选手提交的是基于神经网络的智能体，需要将模型参数文件也放到提交的文件夹下，对模型参数文件的命名规则无要求，只需保证参赛智能体能正常加载即可，每个模型参数文件原则上不超过 100MB。最终提交格式示例如下：

```
|—英文战队名
|    |—__init__.py
|    |—agent.py
|    |—funcs.py
|    |—model.pkl
```

附录 1

2024 年（第二届）“洪都杯”智能空战大赛

Python3.7.X 环境依赖包

序号	依赖库名	版本号	注
1	absl-py	2.0.0	
2	aiosignal	1.3.1	
3	ale-py	0.8.1	
4	astunparse	1.6.3	
5	attrs	23.1.0	
6	cachetools	5.3.2	
7	certifi	2022.12.17	
8	charset-normalizer	3.3.2	
9	click	8.1.7	
10	cloudpickle	2.2.1	
11	colorama	0.4.6	
12	cycler	0.11.0	
13	decorator	5.1.1	
14	distlib	0.3.7	
15	dm-tree	0.1.8	
16	dpcpp-cpp-rt	2024.0.0	
17	easydict	1.1	

18	filelock	3.12.2	
19	filterpy	1.4.5	
20	flatbuffers	2.0.7	
21	flit_core	3.6.0	
22	fonttools	4.38.0	
23	frozenlist	1.3.3	
24	gast	0.4.0	
25	google-auth	2.23.4	
26	google-auth-oauthlib	0.4.6	
27	google-pasta	0.2.0	
28	grpcio	1.51.3	
29	gym	0.26.2	
30	gym-notices	0.0.8	
31	Gymnasium	0.26.3	
32	gymnasium-notices	0.0.1	
33	h5py	3.8.0	
34	idna	3.4	
35	imageio	2.31.2	
36	importlib-metadata	6.7.0	
37	importlib-resources	5.12.0	
38	intel-cmplr-lib-rt	2024.0.0	

39	intel-cmplr-lic-rt	2024.0.0	
40	intel-opencl-rt	2024.0.0	
41	intel-openmp	2024.0.0	
42	joblib	1.3.2	
43	JSBSim	1.1.13	
44	jsonschema	4.17.3	
45	keras	2.7.0	
46	Keras-Preprocessing	1.1.2	
47	kiwisolver	1.4.5	
48	libclang	16.0.6	
49	lz4	4.3.2	
50	Markdown	3.4.4	
51	markdown-it-py	2.2.0	
52	MarkupSafe	2.1.3	
53	matplotlib	3.5.3	
54	mdurl	0.1.2	
55	mkl	2024.0.0	
56	mkl-fft	1.3.1	
57	mkl-random	1.2.2	
58	mkl-service	2.4.0	
59	msgpack	1.0.5	

60	networkx	2.6.3	
61	numpy	1.21.6	
62	oauthlib	3.2.2	
63	opencv-python	4.8.1.78	
64	opt-einsum	3.3.0	
65	packaging	23.2	
66	pandas	1.1.5	
67	Pillow	9.5.0	
68	pkgutil_resolve_name	1.3.10	
69	platformdirs	3.11.0	
70	protobuf	3.20.3	
71	pyasn1	0.5.1	
72	pyasn1-modules	0.3.0	
73	pygame	2.1.0	
74	pyglet	2.0.10	
75	Pygments	2.17.2	
76	pymap3d	2.9.1	
77	pyparsing	3.1.1	
78	pyrsistent	0.19.3	
79	python-dateutil	2.8.2	
80	pytz	2023.3.post1	

81	PyWavelets	1.3.0	
82	PyYAML	6.0.1	
83	ray	2.4.0	
84	requests	2.31.0	
85	requests-oauthlib	1.3.1	
86	rich	13.7.0	
87	rsa	4.9	
88	scikit-image	0.19.3	
89	scipy	1.7.3	
90	setproctitle	1.3.3	
91	simple-pid	2.0.0	
92	six	1.16.0	
93	stable-baselines	2.10.2	
94	tabulate	0.9.0	
95	tbb	2021.11.0	
96	tensorboard	2.11.2	
97	tensorboard-data-server	0.6.1	
98	tensorboard-plugin-wit	1.8.1	
99	tensorboardX	2.6.2.2	
100	tensorflow	2.7.0	
101	tensorflow-estimator	2.7.0	

102	tensorflow-io-gcs-filesystem	0.31.0	
103	tensorflow-probability	0.15.0	
104	termcolor	2.3.0	
105	tifffile	2021.11.2	
106	torch	1.11.0	
107	torchaudio	0.11.0	
108	torchvision	0.12.0	
109	typer	0.9.0	
110	typing_extensions	4.7.1	
111	urllib3	2.0.7	
112	virtualenv	20.21.0	
113	Werkzeug	2.2.3	
114	wincertstore	0.2	
115	wrapt	1.16.0	
116	zipp	3.15.0	