
《物联网与嵌入式技术》

大作业课题报告

基于物联网的无人机对抗强化学习系统 课题报告

学 生 姓 名：_____徐业伟_____

组 号：_____7_____（组长☐ 组员☒）

学 号：_____32403192_____

专 业：_____船舶工程专业_____

完 成 日 期：_____2025.04_____

大连理工大学

Dalian University of Technology

目录

1. 研究背景及设计目标	1
1.1 课题背景	1
1.2 主要设计成果概述	2
1.3 主要设计目标	3
2. 实现过程	4
3. 工作详情	5
3.1 小组分工情况	5
3.2 个人工作	5
4. 结论与致谢	6
附录	7

1. 研究背景及设计目标

1.1 课题背景

近年来，随着无人机技术的快速发展和人工智能的迅速普及，那么将两者进行邮寄的结合便成为了一个逐渐备受关注的研究方向。我们的项目主要关注强化学习领域以及三维仿真平台的研究，结合 VR 技术，实现对空作战的三维场景的复现。

对于强化学习方面的研究，近年来在深度神经网络推动下取得重大突破，DQN、策略梯度等算法已成功应用于机器人控制、游戏 AI 等领域。然而在复杂三维场景尤其是无人机对抗等动态环境中，现有技术面临严峻挑战：传统二维验证环境难以模拟真实空间交互，仿真与现实的鸿沟导致算法性能评估失真；多智能体对抗中的非平稳性、部分可观测性等问题尚未完全解决；Sim-to-Real 迁移中的领域差异和误差累积制约着实际应用。

这些瓶颈凸显了构建高保真三维仿真平台的迫切性。当前研究正从三个维度寻求突破：开发融合注意力机制、记忆模块的新型网络架构以增强空间认知能力；建立精确物理模拟与多样化对抗场景的验证环境；创新课程学习、分层强化等训练范式提升样本效率。本项目正是瞄准这一前沿方向，致力于通过算法与仿真平台的协同创新，解决无人机对抗中的实时决策、战术演化等核心问题，推动强化学习在复杂动态环境中的实际应用。

然而在现阶段的三维仿真平台中，通用引擎的视觉效果与专业工具的物理精度难以兼顾，而面向强化学习的适配性更显不足。Unity、Unreal 等虽提供强大渲染能力，却难以精准模拟无人机对抗所需的空气动力学特性；ANSYS 等专业工具虽物理建模精确，但缺乏对机器学习流程的原生支持。更关键的是，现有平台在提升环境真实性的同时往往牺牲仿真速度，无法满足强化学习海量采样需求，多智能体协同训练所需的分布式交互功能也支持有限。这些瓶颈催生了新一代仿真平台的研发需求——必须实现高保真渲染、精确物理模拟与分布式计算的深度融合，同时构建机器学习专用接口，为智能体训练提供全流程支持。特别是在无人机对抗领域，亟需突破静态环境局限，实现动态地形、可变气象与多机博弈的有机统一，这既是重大技术挑战，也是推动仿真平台向智能化演进的关键机遇。

所以本课题创新新提出构建面向强化学习的高保真空作战仿真平台，建立有效的弥合算法研究与应用落地之间的鸿沟，为强化学习提供更接近真实的训练环境，还将推动智能决策理论在三维复杂场景中的创新发展。研究成果可广泛应用于无人机自主决策、智能空战系统等领域，实现战术演化环境、智能体训练系统与实战验证平台的三位一体，为智能空战系统研发提供完整技术支撑。

1.2 主要设计成果概述

在此次课题项目中，我们设计了土耳其航空工业公司的 TF-X 战斗机、法国达索公司的“阵风”战斗机、通用动力公司的 F-16 战斗机以及 90 年代欧洲战斗机公司生产的 Eurofighter 战斗机，设定了每种战斗机的气动布局、材料、配置等参数，这些参数决定了模拟中战斗机的飞行性能、操控手感和物理真实性。设计了 AIM-120 导弹、MICA 拦截导弹、S-400 防空导弹、Sidewinder 导弹等，对每种导弹的推力、耐久性、伤害、角摩擦等属性进行定义，还考虑了导弹击打过程中的物理特性，使用自定义物理模型对导弹武器轨道进行定义，并在后续通过调整参数来优化导弹性能。

同时实现 ONE ON ONE、ONE ON TWO 等战斗操作页面，实现不同功能的模拟飞行，与 VR 相结合，体验不同的感受。

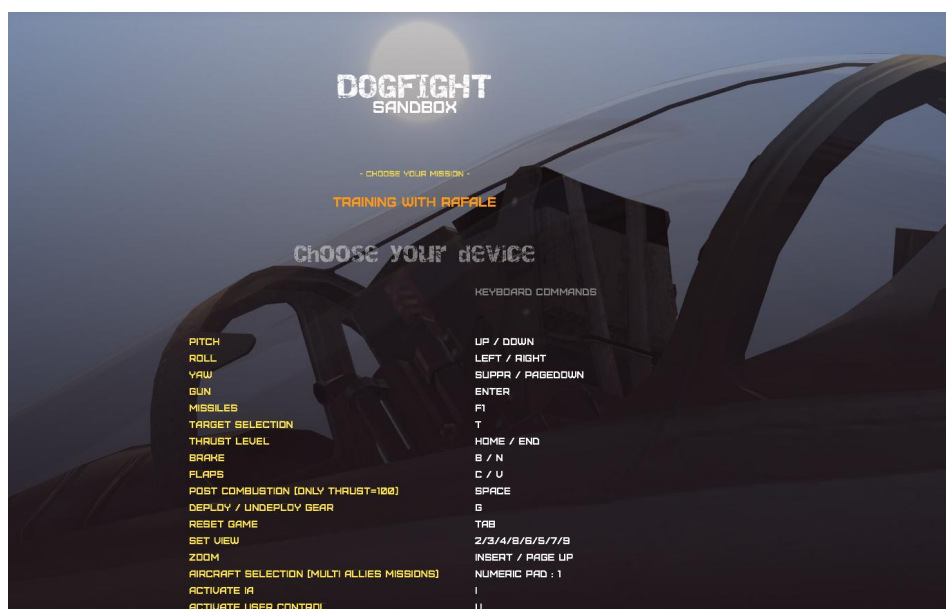


图 1.1 Harfang3D Dog-Fight Sandbox 仿真平台选取页面



图 1.2 ONE ON ONE 实际对抗画面

1.3 主要设计目标

（1）构建一个高保真空战仿真环境

构建一个高保真空战仿真环境，采用航空级动力学模型精确模拟飞行器运动特性。系统基于计算流体力学原理，真实还原飞机在各种飞行状态下的气动特性，包括起飞滑跑时的地面效应、巡航飞行时的稳定控制以及空战机动中的复杂受力情况。环境系统支持动态气象模拟，可呈现不同海拔的云层变化、能见度影响以及风切变等复杂气象条件，同时提供多样化地形地貌，从海平面到高山峡谷，全面考验飞行员的适应能力。通过高精度物理引擎和实时渲染技术，系统能够以 60 帧/秒以上的速率呈现逼真的飞行场景，确保训练过程的流畅性和真实性。

（2）多种控制模式支持：键盘、手柄/摇杆与 VR

在交互方式上，系统提供键盘、专业飞行摇杆和 VR 设备三种控制模式。键盘操作采用智能灵敏度调节算法，使普通键盘也能实现细腻的飞行控制；专业飞行摇杆支持力反馈功能，可真实传递气动载荷变化；VR 模式通过头部追踪和空间定位技术，配合高刷新率显示，打造沉浸式座舱体验。每种控制模式都经过精心调校，确保操作指令的响应延迟低于 50 毫秒，满足专业训练的要求。系统还支持控制方案的完全自定义，用户可以根据个人习惯或特定训练需求调整操作参数。

（3）提供录制、回放与网络联机功能，便于科研与扩展

为支持科研与训练需求，系统提供完善的录制、回放与网络联机功能。训练过程的所有关键数据，包括飞行参数、操作指令和环境状态等，都可以被完整记录并以多种格式导出，便于后续分析研究。智能回放系统支持多视角观察和参数曲线叠加显示，帮助用户深入分析每一个战术动作。网络联机功能采用优化的数据传输协议，在保证同步精度的同时将网络延迟控制在 100 毫秒以内，最多支持 16 机同时在线对抗。系统还提供开放的 API 接口，方便研究人员接入自主开发的智能算法或扩展新的功能模块。

这三个核心功能模块相互配合，共同构成一个完整的空战仿真生态系统。高保真环境为训练提供真实场景，多种控制模式满足不同用户需求，而完善的录制和联机功能则为系统赋予了强大的科研扩展能力。通过持续优化和迭代，该平台将成为航空训练和战术研究的重要工具。

2. 实现过程

“Dogfight 空战沙盘”项目以 Python 3 作为开发语言，Harfang3D 2 为核心框架，全力打造一个功能完备、高度拟真的空战仿真环境。

在整体架构设计层面，项目规划了多个功能模块。输入管理模块负责精准采集和处理来自不同设备的操作指令，确保玩家的每一个动作都能被系统准确识别；物理仿真模块运用算法，对飞机、导弹等各类物体的运动进行模拟，从飞行轨迹到碰撞效果，还原真实物理规律；图形渲染模块借助 Harfang3D 2 强大的图形处理能力，结合内置着色器，呈现海洋、地形、天穹、云层等丰富场景元素，为玩家带来身临其境的视觉感受；音效模块通过动态调节技术，依据飞行状态实时变化音效，如飞机加速时引擎轰鸣声增大，让玩家仿佛置身于激烈的空战战场；录制回放模块则为玩家和科研人员提供了记录和复盘战斗过程的便利，有助于总结经验和分析战术。

同时，项目采用状态机架构，将游戏进程划分为菜单状态、主游戏状态、回放状态和结束状态。在菜单状态下，玩家可以进行游戏设置、选择模式等操作；主游戏状态中，各种游戏逻辑和交互功能全面展开；回放状态方便玩家回顾精彩瞬间；结束状态则对游戏结果进行展示和处理。每个状态都配备了专门的初始化函数和按帧更新函数，初始化函数负责在状态切换时进行必要的准备工作，按帧更新函数则确保在每一帧画面中，状态内的各种元素都能根据游戏进展进行相应的变化，从而保证游戏的流畅性和连贯性。

在物理仿真与动态效果的打造上，对于飞机、导弹等对象的物理运动，采用高精度动力学计算，充分考虑空气阻力、重力、推力等多种因素，使它们在虚拟空间中的飞行轨迹和动作表现符合现实规律。在飞机起飞、降落和机动过程中，配合细腻的物理反馈，比如起飞时机身的震动、降落时的着地感，让玩家切实感受到真实飞行的体验。ParticlesEngine 模块发挥着关键作用，它能够生动地模拟烟雾、火焰、爆炸等环境特效，为战斗场景增添了强烈的视觉冲击力；Destroyable_Machine 模块则专注于碰撞检测和目标销毁反馈，当飞机或导弹发生碰撞时，精确判断碰撞结果，并根据实际情况呈现目标被摧毁的效果，极大地增强了游戏的交互打击感。在音效方面，通过先进的音频处理技术，根据飞行状态动态调节音效，实现平滑渐变和精准的 3D 音频定位，同时在环境中添加逼真的背景音、引擎声以及碰撞反馈音效，进一步提升游戏的沉浸感。

在关键技术实现上，HARFANG 3D 引擎作为核心图形处理工具，支持 DirectX 和 OpenGL 两种主流图形接口，能够根据不同硬件环境进行优化适配。利用其内置的高级着色器技术，项目实现了复杂的光影效果，如环境光遮蔽让场景中的物体在阴影处更加真实自然，动态阴影随着物体的运动实时变化，大幅提升了视觉体验。同时，高级着色器技术还对渲染效率进行了优化，通过合理分配图形计算资源，减少不必要的计算负担，确保在复杂场景下，游戏仍能保持流畅的帧率。此外，项目还借助该引擎的功能，模拟真实气象条件，包括云层的飘动、雾气的弥漫、雨雪的降落等，增加了空战模拟的复杂

性和挑战性；利用精细的地形地貌生成技术，创建出山脉、平原、海洋等多样化的战场环境，为玩家制定战术提供了更多可能性。

此外，项目兼容 SteamVR，支持 HTC Vive、Oculus Rift 等主流 VR 设备。在 VR 技术集成过程中，项目团队对 VR 设备进行了特殊优化，通过降低延迟和优化画面渲染等措施，确保玩家在 VR 环境下也能获得流畅、逼真的空战体验，仿佛真正置身于激烈的空战战场之中。

3. 工作详情

3.1 小组分工情况

本组分工情况如下所示：

飞行器设计：俞越洋，王健羽，候若渝，张庆熙

武器设计：黄奕博，徐业伟，顾建强，钟鹏

强化学习系统算法设计：从金淼，许芷毓

虚拟现实技术实现：孟祥驰

3.2 个人工作

在此次课题研究中，我主要负责武器系统的设计，主要编译 MICA 导弹的相关工作。MICA 导弹全称为“拦截与空战导弹”（Missile d'Interception et de Combat Aerien），是法国马特拉公司于 80 年代自行研制的第四代空空导弹，用于取代“马特拉”超 530F/D 中距拦射导弹和 R550 “魔术” 2 近距格斗导弹，1997 年开始服役，目前已被法国达索公司“阵风”战斗机选用。

通过网上开源资料，搜索到 MICA 相关数值资料；其弹长 3.1 米，弹径 0.165 米，翼展 0.61 米，弹重 112 千克，战斗部仅 12 千克。图 2.1 为其结构示意图。

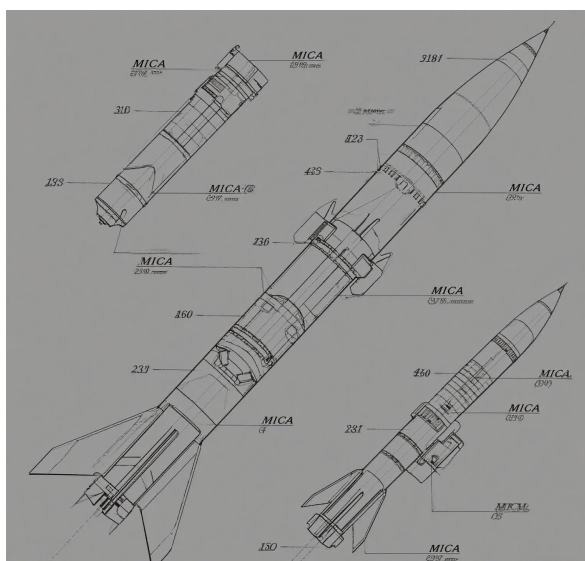


图 2.1 MICA 结构示意图

通过开源得到的示意图，结合具体物理学公式等，研究建立几何模型，并设置了导弹推力、烟雾效果、空气阻力等物理参数，符合游戏引擎中飞行器的常规属性配置逻辑。使用均匀随机数模拟伤害浮动，符合游戏化设计需求。其具体相关代码见附录所示。

$$\vec{F}_{move} = \vec{F}_{thrust} + \vec{F}_{lift} - \vec{F}_{drag} + \vec{F}_{gravity}$$

4. 结论与致谢

在本学期研究生课程项目中，我们团队倾力打造的高保真空战仿真系统研发工作已圆满收官。这一融合多学科前沿技术的创新实践，不仅让我们收获了丰硕的技术成果，更完成了一次从理论到实践的科研淬炼。

项目研发过程中，我们突破了多项关键技术瓶颈。基于有限体积法构建的简化 CFD 模型，在保证实时性的同时实现了升力系数、阻力系数等关键气动参数的精确模拟；采用 Unity3D 引擎开发的虚拟战场环境，通过程序化地形生成和动态天气系统，生动再现了复杂多变的空战场景；自主设计的网络同步框架，运用状态同步与预测回滚机制，确保了多机对抗的流畅体验。这些技术突破让我们深刻体会到工程实践中平衡精度与效率的艺术。

在项目推进过程中，我们得到了多位老师的专业指导与支持。首先要特别感谢各位老师和助教的帮助，在实验研究器材以及思路方面的悉心指导，帮助我们克服了关键的技术障碍，为我们解决了数据传输的难题。还要感谢实验室提供的计算资源支持，为项目开发创造了良好的硬件环境。

这个项目能够顺利完成，离不开团队每位成员的共同努力。大家克服课业压力，投入大量时间进行系统开发和调试，展现了出色的专业素养和团队精神。虽然作为课程项目已告一段落，但我们认识到系统在气动模型精度、网络同步算法和环境真实感等方面仍有提升空间。这次实践不仅让我们掌握了专业技能，更激发了继续深入研究的兴趣，为未来的科研工作奠定了坚实基础。我们期待能在相关领域开展更深入的研究，将课程所学转化为更有价值的科研成果。

附录

Rafale 与 MICA 相关参数代码

```

import harfang as hg
from Machines import *
class Rafale_Parameters:
    model_name = "Rafale"
    instance_scene_name = "machines/rafale/rafale_rigged.scn"
    def __init__(self):
        # Aircraft constants:
        self.camera_track_distance = 30
        self.thrust_force = 15
        self.post_combution_force = self.thrust_force / 2
        self.drag_coeff = hg.Vec3(0.043, 0.07666, 0.0003)
        self.wings_lift = 0.0005
        self.brake_drag = 0.006
        self.flaps_lift = 0.0025
        self.flaps_drag = 0.002
        self.angular_frictions = hg.Vec3(0.000165, 0.000115, 0.000255) # pitch, yaw,
roll
        self.speed_ceiling = 2200 # maneuverability is not guaranteed beyond this
speed !
        self.angular_levels_inertias = hg.Vec3(3, 3, 3)
        self.max_safe_altitude = 15240
        self.max_altitude = 25240 * 4
        self.gear_height = 2.28
        self.bottom_height = 0.78
        # Weapons configuration:
        self.missiles_config = ["Mica", "Meteor", "Meteor", "Meteor", "Meteor",
"Mica"
        # Mobile parts:
        self.mobile_parts_definitions = [
            ["aileron_left", -20, 20, 0, "dummy_rafale_wing_flap_l", "X"],
            ["aileron_right", -20, 20, 0, "dummy_rafale_wing_flap_r", "X"],
            ["elevator_left", -45, 45, 0, "dummy_rafale_elevator_l", "X"],
            ["elevator_right", -45, 45, 0, "dummy_rafale_elevator_r", "X"],
            ["rudder", -30, 30, 0, "rafale_rudder", "Y"]
        ]
    ]
class Rafale(Aircraft, Rafale_Parameters):
    @classmethod
    def init(cls, scene):
        print("Rafale class init")

```

```

    def __init__(self, name, scene, scene_physics, pipeline_ressource:
hg.PipelineResources, nationality, start_pos, start_rot):
        self.gear_anim_play = None
        Aircraft.__init__(self, name, Rafale_Parameters.model_name, scene,
scene_physics, pipeline_ressource, Rafale.instance_scene_name, nationality, start_pos,
start_rot)
        Rafale_Parameters.__init__(self)
        self.define_mobile_parts(self.mobile_parts_definitions)
        self.add_device(Gear("Gear", self, scene, self.get_animation("gear_open"),
self.get_animation("gear_fr_close")))
        self.setup_bounds_positions()
    def update_mobile_parts(self, dts):
        self.parts["aileron_left"]["level"] = -self.angular_levels.z
        self.parts["aileron_right"]["level"] = self.angular_levels.z
        self.parts["elevator_left"]["level"] = -self.angular_levels.x
        self.parts["elevator_right"]["level"] = -self.angular_levels.x
        self.parts["rudder"]["level"] = -self.angular_levels.y
        Aircraft.update_mobile_parts(self, dts)

import harfang as hg
from random import uniform
from Machines import *
class Mica(Missile):
    model_name = "Mica"
    instance_scene_name = "weaponry/missile_mica.scn"
    @classmethod
    def init(cls, scene):
        print("Mica missile class init")
    def __init__(self, name, scene, scene_physics, pipeline_ressource:
hg.PipelineResources, nationality):
        Missile.__init__(self, name, Mica.model_name, nationality, scene,
scene_physics, pipeline_ressource, Mica.instance_scene_name)
        self.f_thrust = 150
        self.smoke_parts_distance = 1.44374
        self.angular_frictions = hg.Vec3(0.00014, 0.00014, 0.00014) # pitch, yaw, roll
        self.drag_coeff = hg.Vec3(0.37, 0.37, 0.0003)
        self.life_delay = 15
        self.smoke_delay = 1
    def get_hit_damages(self):
        return uniform(0.20, 0.30)

```