# 从概念到数字孪生：城市无人机集成仿真工作流的可行性与工具链分析

## 执行摘要与关键发现

本报告旨在对一个集成了世界构建、物理预测、路径规划、飞控仿真与可视化重建的“数字孪生城市”无人机试飞研究项目流程进行深入的可行性分析与工具链评估。该项目设想具有典型的跨学科特征，其成功与否高度依赖于不同专业模块之间数据传递的无缝衔接。通过对每个“传递”环节进行详尽的调查，本报告旨在打破“纸上谈兵”的现状，为项目的实际落地提供一份详尽、可操作的技术路线图。

分析表明，该项目设想在技术上是可行的，但其复杂性与实施难度远超表面所见。多个关键环节并非简单的文件格式转换，而是需要复杂的数据预处理、几何修复、坐标系变换以及定制化的脚本开发。其中，从三维城市模型到计算流体动力学（CFD）可用几何体的转换过程是整个工作流中最大的技术瓶颈与工作量所在。此外，在项目启动之初建立一个统一的坐标参考系统，是确保所有空间数据（地理模型、风场、飞行轨迹）能够正确对齐的根本前提，也是原始设想中被忽略的关键步骤。

下表总结了整个工作流中各数据传递环节的核心任务、所需工具及可行性评估，旨在为项目规划者提供一个宏观的参考框架。

**表 1：数据传递与转换矩阵**

| 步骤传递 | 源模块与输出 | 目标模块与所需输入 | 核心转换工具/方法 | 传递性质 | 可行性与工作量评估 |
| --- | --- | --- | --- | --- | --- |
| **1 → 3** | **世界构建**(GIS/AIGC)输出：.obj | **物理预测 - CFD**(ANSYS Fluent)输入: | ANSYS SpaceClaim | 手动几何修复、简化与 | **可行，但工作量巨大。**此为项目最 |

| | | | | | |
|---|---|---|---|---|---|
| | / .fbx | 封闭、简化的流体域几何 | | 流体域构建 | 关键的瓶颈，需要专业的 CAD 处理技能。 |
| **2 → 3** | **物理预测 - 气象** (WRF 等)输出: .nc (NetCDF) | **物理预测 - CFD**(ANSYS Fluent)输入: 边界条件剖面文件 (.prof) | 定制化 Python 脚本 (xarray, wrf-python 库) | 脚本化数据提取、插值与格式化 | **可行，中等工作量**。需要编写专用脚本将气象数据转换为 Fluent 可读的格式。 |
| **3 → 4** | **物理预测 - CFD**(ANSYS Fluent) 输出: .vtk | **路径规划**(算法模块) 输入: 任意点风场矢量查询 | 定制化 Python 脚本 /API (PyVista 库) | 将风场数据封装为可查询的服务 | **可行，中等工作量**。需要将静态 VTK 文件转化为一个动态查询接口。 |
| **4 → 5** | **路径规划**(算法模块) 输出: .csv | **PX4 仿真** (PX4 SITL/Gazebo) 输入: MAVLink 任务协议 | MAVSDK-Python 库 | 脚本化任务生成与协议上传 | **可行，中等工作量**。原始设想的.csv 传递方式不适用，需通过 MAVSDK 进行编程实现。 |
| **5 → 6** | **PX4 仿真** (PX4 SITL)输出: .ulg (飞行日志) | **AIGC 可视化** (渲染引擎) 输入: 三维轨迹与姿态数据 | pyulog Python 库 | 脚本化日志解析与数据提取 | **可行，低等工作量**。pyulog 库提供了成熟的日志文件解 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 析方案。 |
| **1,3,5 → 6** | **多模块**输出: .fbx, .vtk, .csv | **AIGC 可视化**(渲染引擎)输入: 场景、流场、轨迹 | NVIDIA Omniverse, Unreal Engine, Unity | 多源数据集成与实时渲染 | **可行，中到高工作量**。取决于所选平台对科学数据的支持程度，Omniverse 是首选。 |

综上所述，该项目需要一支具备复合技能的团队，不仅要精通各自领域的专业软件，还必须具备强大的系统集成和编程能力，特别是 Python 脚本开发能力，以构建连接各个独立模块的数据桥梁。本报告将逐一剖析这些挑战，并提供详实的技术解决方案。

# 第一部分：基础性挑战——统一坐标系

在启动任何数据生成或处理步骤之前，必须解决一个在原始设想中被忽略、却至关重要的基础性问题：建立一个贯穿整个项目的统一坐标参考系统（Coordinate Reference System, CRS）。若无此前提，GIS 生成的城市模型、CFD 计算的风场以及 PX4 仿真的无人机轨迹将处于相互独立的坐标空间中，最终的可视化结果将是毫无意义的错位拼贴。

## 各模块迥异的参考框架分析

项目中涉及的软件工具源于不同领域，其默认的坐标系约定也大相径庭。

- **GIS（步骤 1）**：无论是通过 ArcGIS 还是 Blender-OSM 等工具，获取的地理空间数据通常基于地理坐标系（Geographic Coordinate System, GCS），如 WGS 84，使用经度、纬度和海拔高度来定义位置[1]。这些是定义在地球椭球体上的全局或大尺度坐标。
- **CFD（步骤 3）**：ANSYS Fluent 等 CFD 软件在纯粹的局部笛卡尔坐标系（X, Y, Z）中运行 。它没有地理参考的概念，所有几何模型都必须以米为单位，相对于一个任意设定的局部原点（0,0,0）进行定义。
- **PX4/Gazebo（步骤 5）**：PX4 飞控固件和 Gazebo 仿真环境广泛采用局部切平面坐标系（Local Tangent Plane），通常是**东北天（ENU - East, North, Up）**或**北东地（NED - North, East, Down）**坐标系[4]。这两种坐标系的关键区别在于 Z 轴的方向：ENU 中 Z 轴向上为正，而 NED 中 Z 轴向下为正。大多数 3D 建模和仿真软件（包括 Gazebo）默认 Z 轴向上，与 ENU 习惯一致，而航空航天领域则更常使用 NED 约定[5]。

这种坐标系的不统一是项目集成的首要障碍。若采用被动的、步步转换的策略（例如，先将 GIS 数据导入 Fluent 的任意坐标系，再尝试将其与 Gazebo 的 ENU/NED 对齐），将极易引入累积误差和方向混淆，导致整个仿真链的物理真实性失效。因此，必须采取主动的、前置的策略，在项目之初就定义一个所有模块都必须遵守的“黄金标准”坐标系。

## 统一参考框架的规定性策略

为确保所有空间数据的一致性，兹推荐以下三步走的策略来建立一个统一的项目级坐标参考框架。

1. **定义局部原点**：在 GIS 数据中选择一个明确的、固定的地理点作为整个仿真世界的绝对原点（0,0,0）。例如，可以选择广州市模型区域的几何中心点。这个点拥有唯一的经度、纬度和海拔高度，它将成为所有局部坐标的参考基准。
2. **建立局部切平面**：将项目的仿真空间定义为一个**东北天（ENU）**坐标系。选择 ENU 框架的主要优势在于其 Z 轴向上的约定与大多数 3D 建模软件（如 Blender）、CFD 前处理器（如 SpaceClaim）以及仿真器（如 Gazebo）的默认视口和操作习惯保持一致，这能最大限度地减少在工具间切换时的方向混淆和操作失误[4]。
3. **实施坐标转换逻辑**：使用 Python 的 pyproj 库作为所有坐标转换任务的标准化工具。pyproj 是 PROJ 库的 Python 接口，提供了强大而精确的地理坐标转换功能[6]。需要编写一个核心的工具脚本，该脚本能够将任何来自 GIS 数据的"经度、纬度、高度"坐标，精确转换为相对于已定义局部原点的 ENU 坐标（东向为 X，北向为 Y，天向为 Z）。

以下是一个 Python 代码示例框架，演示了如何使用 pyproj 实现这一核心转换功能：

Python

```
import pyproj

class CoordinateTransformer:
    def __init__(self, origin_lat, origin_lon, origin_alt):
        """
        初始化坐标转换器，定义局部 ENU 坐标系的原点。
        :param origin_lat: 原点的纬度 (WGS 84)
        :param origin_lon: 原点的经度 (WGS 84)
        :param origin_alt: 原点的高度 (米)
        """
        # 定义全局地理坐标系 (WGS 84)
        self.geodetic_crs = pyproj.CRS("EPSG:4326") # WGS 84 Lat/Lon

        # 定义地心坐标系 (ECEF)
        self.ecef_crs = pyproj.CRS("EPSG:4978") # ECEF (Earth-Centered, Earth-Fixed)

        # 定义局部 ENU 坐标系
        # PROJ 字符串定义了一个以 origin_lon, origin_lat 为中心的局部切平面坐标系
```

```
        proj_string = (
            f"+proj=etmerc +ellps=WGS84 +lat_0={origin_lat} +lon_0={origin_lon} "
            f"+x_0=0 +y_0=0 +k_0=1 +units=m +vunits=m +no_defs"
        )
        self.local_enu_crs = pyproj.CRS.from_proj4(proj_string)

        # 创建从地理坐标到局部 ENU 坐标的转换器
        self.geo_to_enu_transformer = pyproj.Transformer.from_crs(
            self.geodetic_crs, self.local_enu_crs, always_xy=True
        )

    def geo_to_enu(self, lat, lon, alt):
        """
        将单个地理坐标点转换为局部 ENU 坐标。
        注意：此基本转换忽略了高程。对于高精度应用，需要更复杂的 ECEF 转换。
        一个更严谨的方法是先转为 ECEF，再转为 ENU。
        """
        # pyproj 的 transform 方法主要处理 2D 平面投影
        # 高程(Z)通常需要分开处理
        x, y = self.geo_to_enu_transformer.transform(lon, lat)
        z = alt - origin_alt # 简化的处理方式
        return x, y, z

# --- 使用示例 ---
# 1. 定义广州市某区域中心的经纬高作为原点
origin_lat = 23.1291
origin_lon = 113.2644
origin_alt = 10.0 # 假设的海拔高度

# 2. 创建转换器实例
transformer = CoordinateTransformer(origin_lat, origin_lon, origin_alt)

# 3. 转换一个 GIS 数据点
point_lat = 23.1300
point_lon = 113.2650
point_alt = 50.0

enu_x, enu_y, enu_z = transformer.geo_to_enu(point_lat, point_lon, point_alt)

print(f"地理坐标 ({point_lat}, {point_lon}, {point_alt}) 转换为 ENU 坐标:
({enu_x:.2f}, {enu_y:.2f}, {enu_z:.2f}) 米")
```

这个转换器（或其变体）将成为项目中的一个基础工具，确保从城市模型顶点、WRF 气象数据网

格点，到无人机任务航点的所有空间信息，都能被无歧义地表示在同一个三维空间中，为后续所有模块的集成奠定坚实基础。

# 第二部分：从世界模型到仿真可用几何（步骤 1 → 3）

此环节涉及将步骤 1 中由世界构建模块（如 GIS 程序或 AIGC 技术）生成的 3D 城市模型，传递给步骤 3 的 CFD 模块（ANSYS Fluent）。这是整个工作流中可行性挑战最大、最易被低估的环节，其本质并非简单的文件导入，而是一个劳动密集型的几何重构过程。

## 可行性评估：一个关键的技术瓶颈

将.obj、.stl 或.fbx 等格式的 3D 模型导入 ANSYS Fluent 是可行的，但这仅仅是万里长征的第一步[8]。从 GIS 或 AIGC 工具直接导出的原始模型，由于其几何特性，完全不满足 CFD 分析的严苛要求，直接使用将导致网格划分失败或计算结果毫无意义。

- **CFD 对几何的核心要求**：CFD 分析的对象是流体本身，而非固体。因此，提供给 CFD 软件的几何模型必须是代表流体流动空间的“流体域”。这个流体域必须是一个“水密”（watertight）的、封闭的、无自相交的单一实体。此外，为了控制计算成本，几何模型必须经过简化（defeaturing），去除所有对宏观流动影响甚微的微小细节，如窗框、门把手、天线等[11]。
- **GIS/AIGC 模型的固有缺陷**：无论是基于 GIS 数据程序化生成的城市模型，还是通过 AIGC 技术创建的虚拟城市，其产出物通常是为视觉表现而优化的。这些模型往往是“多边形汤”（polygon soup），即大量独立的、不连接的多边形面的集合，充满了孔洞、缝隙、重叠面和非流形边[15]。AIGC 生成的模型，尽管在视觉上可能更复杂和逼真，但其几何拓扑结构可能更加混乱，包含更多的非流形结构和内部交叉，这使得几何清理工作变得更为艰巨[11]。

因此，这个传递环节的本质，是从一个“视觉模型”到一个“分析模型”的转变。这中间的鸿沟必须通过一个专门的几何处理工具来填补，而这个过程需要大量的人工干预和专业判断。

## 工具链深度解析：ANSYS SpaceClaim 的角色与工作流

在 ANSYS 生态系统中，ANSYS SpaceClaim 是专为此类几何准备任务而设计的直接建模工具。它充当了原始 CAD/Mesh 数据与 Fluent 网格划分模块之间不可或缺的桥梁[8]。以下是在 SpaceClaim

中处理城市模型的标准工作流程：

1. **导入模型**：启动 SpaceClaim，将生成的城市模型文件（如.obj 或.stl）导入到工作区[8]。
2. **几何修复**：这是最关键的一步。利用 SpaceClaim 的“修复”工具栏，执行一系列操作以形成封闭的实体。
   - **缝合（Stitch）**：自动检测并缝合相邻曲面之间的微小间隙，将离散的曲面片组合成一个整体[22]。
   - **间隙（Gaps）**：高亮显示模型中大于指定公差的缝隙，以便手动修补。
   - **缺失面（Missing Faces）**：通过选择边界边线，自动创建缺失的曲面来封堵孔洞。
   - 对于复杂的城市模型，这一过程可能需要反复迭代，并结合手动创建曲面来处理大型或不规则的孔洞。
3. **几何简化（Defeaturing）**：为了降低后续网格划分的复杂度和计算量，必须去除不必要的细节。
   - **填充（Fill）**：这是一个极其强大的工具，只需选择要移除的特征（如窗户凹陷、阳台、小型凸起）的边界边线，即可快速将其“填平”，生成光滑的表面[14]。
   - **强力选择（Power Selection）**：可以智能地选择所有具有相似几何特征（如半径相同的圆角、面积相同的孔洞）的面，然后一次性使用“填充”工具将其全部移除[20]。
   - **收缩包覆（Shrinkwrap）**：作为一种快速但较为粗糙的方法，可以创建一个包裹在原始几何外部的、水密的、简化的外壳。对于极其复杂的模型，这可以作为初步简化的第一步[14]。
4. **实体合并**：将所有独立的、已修复和简化的建筑实体，通过“合并（Combine）”工具联合成一个单一的实体（Solid Body）。
5. **创建流体域**：
   - **外壳（Enclosure）**：在“准备”选项卡下，使用“外壳”工具在合并后的城市实体周围创建一个足够大的长方体。这个长方体代表了我们关心的空气计算域[8]。
   - **布尔减法**：使用“合并”工具，从外壳长方体中减去城市实体。操作完成后，剩下的部分即为流体域——一个包含了建筑形状空腔的单一空气实体。
6. **命名选择（Named Selections）**：这是确保边界条件正确施加的关键步骤。在 SpaceClaim 中，为流体域的各个外表面创建“命名选择”。例如，将迎风面命名为 inlet，背风面命名为 outlet，顶部命名为 top，两侧命名为 sides，地面和建筑表面命名为 walls。这些名称会被 Fluent 自动识别，极大地简化了边界条件的设置过程[26]。
7. **共享拓扑**：完成上述步骤后，将清理好的几何体发送到 ANSYS Workbench 中的 Fluent Meshing 模块。在 Workbench 层面，确保几何体单元与 Fluent 单元之间的拓扑共享，使得命名选择等信息可以正确传递。

这个过程将一个不适合分析的视觉资产，转化为一个结构良好、拓扑正确、为 CFD 计算量身定制的分析模型。低估这一步骤所需的时间、精力和专业技能，是导致此类集成项目失败的主要原因之一。项目规划者必须认识到，这并非一个可以完全自动化的“转换”按钮，而是一个需要投入大

量工程时间的“重塑”过程。

# 第三部分：耦合中尺度气象与微尺度 CFD（步骤 2 → 3）

此环节的目标是将由中尺度气象模型（如 WRF）生成的城市上空大尺度气象数据，作为边界条件输入到微尺度 CFD 仿真（ANSYS Fluent）中。这是一种在城市气象学和风工程领域广泛应用的单向耦合方法，在技术上是完全可行的，并有大量学术研究作为支撑 [28]。然而，实现这一传递需要克服数据结构、空间分辨率和坐标系统之间的“数据阻抗不匹配”问题，核心在于开发一个定制化的数据处理脚本。

## 可行性评估：标准但需定制的耦合流程

WRF 模型输出的 wrfout_*.nc 文件包含了随时间和空间变化的三维气象变量（风速、温度、压力等），而 ANSYS Fluent 则需要在其计算域的边界（如入口面）上定义这些变量的分布。直接的文件格式转换是行不通的。必须通过一个中间脚本来提取、插值和格式化数据。

## 工具链深度解析：构建 WRF 到 Fluent 的 Python 桥接脚本

一个功能完备的 Python 脚本是实现此环节数据传递的关键。该脚本将扮演数据转换器的角色，其开发流程可分为以下三个步骤：

1. **解析 WRF NetCDF 输出文件**：
    - **核心库**：netCDF4 是读取 NetCDF 文件的底层库，而 xarray 则提供了更高级、带标签的数据结构（类似于 Pandas DataFrame），使数据操作更为直观 [33]。wrf-python 是一个专门为 WRF 输出设计的库，它封装了许多常用的诊断计算和插值函数，能极大地简化数据提取过程 [37]。
    - **脚本逻辑**：
        1. 使用 netCDF4.Dataset 打开 wrfout_*.nc 文件。
        2. 利用 wrf.getvar()函数，为指定的模拟时间点提取所需的气象变量场，例如 U（西风分量）、V（南风分量）、W（垂直风分量）、T（温度）以及湍流相关变量（如湍动能 TKE）的三维数组 [37]。

3. 同时，从文件中提取对应的地理坐标信息（经度 XLONG，纬度 XLAT）和垂直高度坐标。

2. **生成 Fluent 边界剖面文件（Profile File）**：
   - **Fluent Profile 文件**：ANSYS Fluent 支持从一种特殊格式的文本文件（通常以.prof 为后缀）中读取非均匀的边界条件 [39]。这种方法避免了编写和编译复杂的 UDF（User-Defined Function）的需要 [42]。
   - **文件格式**：该文件采用一种类似 `Lisp` 的语法，以嵌套括号组织数据。每个剖面（`profile`）由一个头部和若干个字段（`field`）组成。头部定义了剖面名称、类型（对于空间变化的边界条件，通常使用 `point` 类型）和数据点数量。每个字段包含字段名（如坐标 x, y, z 或物理量 `u-velocity`, `temperature`）和一列对应的值。所有数值都必须使用 **SI 单位** [43]。
   - **脚本逻辑（续）**：
     1. 根据 CFD 模型的尺寸和位置，在第一部分建立的统一 ENU 坐标系中，确定入口边界面的几何范围。
     2. 在入口面上生成一个采样点网格。这些点的坐标将作为 Profile 文件中的 x, y, z 字段。
     3. 对于每个采样点，将其 ENU 坐标转换回地理坐标（经纬高）。
     4. 使用三维插值方法（例如 `scipy.interpolate.griddata`），从 WRF 的粗网格数据中，插值得到该地理坐标点上的风速分量、温度等气象变量值。
     5. 将插值得到的所有物理量（如 `u-velocity`, `v-velocity`, `w-velocity`, `temperature`, `turbulent-kinetic-energy`, `turbulent-dissipation-rate` 等）组织成与坐标点一一对应的列表。
     6. 按照 Fluent Profile 文件的严格语法，将剖面名称、数据点数量、所有坐标字段和物理量字段格式化成一个长字符串。
     7. 将该字符串写入一个文本文件，例如 `inlet_boundary.prof`。

   下面是一个.prof 文件的示例结构：

```
Scheme
((inlet_profile point 4)
 (x 0.0 0.0 0.0 0.0)
 (y 0.0 0.0 10.0 10.0)
 (z 0.0 100.0 0.0 100.0)
 (u-velocity 10.1 12.5 10.2 12.6)
 (v-velocity 1.1 1.5 1.2 1.6)
 (temperature 298.15 297.50 298.10 297.45)
)
```

3. **在 Fluent 中导入并应用 Profile**：
   - 在 Fluent 的边界条件设置面板中，选择入口面（例如，之前命名为 `inlet` 的表面）。
   - 对于风速、温度等需要设置的参数，选择下拉菜单中的 `profile` 选项，而不是输入一个常数值。
   - 通过菜单 `File -> Read -> Profile...`，选择并加载刚刚由 Python 脚本生成的.prof

文件[44]。

- Fluent 会自动读取文件中的数据，并根据文件内定义的坐标点，将边界条件值插值到入口面的每一个网格单元上。

通过这个定制化的脚本化流程，中尺度气象模型的宏观预测结果被有效地“降尺度”并应用为微尺度 CFD 仿真的驱动力，确保了仿真初始条件的物理真实性。这个桥接脚本是连接两个不同尺度物理模型的关键枢纽，其实现的准确性直接影响最终风场预测的质量。

# 第四部分：从流体动力学到可操作情报（步骤 3 → 4）

此环节负责将 ANSYS Fluent 计算出的局部三维风场数据（city_windfield.vtk），传递给路径规划模块。这是一个将纯物理仿真结果转化为决策依据的关键步骤。该传递过程可行性高，因为 VTK（Visualization Toolkit）是一种广泛应用于科学计算可视化的标准数据格式，拥有成熟的开源库支持。

## 可行性评估：从静态数据到动态查询

路径规划算法（无论是 A*、强化学习还是模型预测控制）的本质是一个探索或优化的过程。它不会一次性加载和处理整个可能高达数 GB 的 VTK 风场文件。相反，在其算法执行的每一步，它都需要动态地查询环境信息，即提出类似“在空间中的某一个特定点(x, y, z)，风速矢量是多少？”这样的问题。因此，这个传递环节的核心任务，是将静态的 VTK 文件转化为一个能够响应实时查询的服务接口。

## 工具链深度解析：利用 PyVista 构建风场查询 API

PyVista 是一个现代化的、高级的 Python 库，它封装了底层的 VTK 库，提供了极其便利的接口来读取、处理和可视化网格数据[45]。它是在路径规划模块中实现风场查询功能的理想工具。

1. **读取 VTK 风场数据**:
   - PyVista 可以一行代码读取.vtk 文件，并将其中的网格几何、节点坐标和附带的物理量（如速度矢量）加载到一个 DataSet 对象中[46]。
   - **代码示例**:
     ```Python
     import pyvista as pv

     # 加载 Fluent 导出的 VTK 文件
     wind_field_grid = pv.read('city_windfield.vtk')
     ```

```
# 假设风速矢量数据在 VTK 文件中被命名为'velocity'
# 检查数据是否成功加载
print("VTK 文件加载成功。")
print("包含的数据场:", wind_field_grid.point_data.keys())
```

2. **实现按需矢量插值（核心功能）**：
   - 路径规划器需要在任意空间点获取风速，而这些点很可能不在 VTK 文件的网格顶点上。因此，必须进行插值计算。
   - PyVista 的 DataSet 对象提供了.interpolate()或.sample()方法。这些方法能够接收一个包含任意探测点坐标的目标网格，并利用源网格（风场网格）的数据，在这些探测点上插值计算出相应的物理量值[48]。
   - **代码示例**：构建一个核心查询函数 get_wind_at_points。该函数将成为风场数据和路径规划算法之间的标准接口（API）。

```
Python
import numpy as np
import pyvista as pv

# (续上文) 假设 wind_field_grid 已加载

def get_wind_at_points(query_points: np.ndarray):
    """
    查询并插值获取指定三维空间点上的风速矢量。

    :param query_points: 一个 Numpy 数组，形状为(N, 3)，每行是一个[x, y, z]坐标。
    :return: 一个 Numpy 数组，形状为(N, 3)，每行是对应点的[u, v, w]风速矢量。
    """
    if not isinstance(query_points, np.ndarray) or query_points.ndim!= 2 or query_points.shape!= 3:
        raise ValueError("输入必须是形状为(N, 3)的 Numpy 数组。")

    # 将查询点封装为 PyVista 的 PolyData 对象
    probe_locations = pv.PolyData(query_points)

    # 使用 interpolate 方法进行插值
    # 'sharpness'和'radius'参数可以调整插值核
    interpolated_data = probe_locations.interpolate(wind_field_grid, sharpness=2.0)

    # 从插值结果中提取名为'velocity'的数据场
    # 假设风速矢量在 VTK 文件中名为'velocity'
```

```
    wind_vectors = interpolated_data.point_data['velocity']

    return wind_vectors

# --- 使用示例 ---
# 假设路径规划算法需要评估三个候选点的风况
candidate_points = np.array([100.5, 250.2, 50.0],
    [110.0, 255.0, 55.1],
    [120.8, 260.5, 60.3])

wind_vectors_at_points = get_wind_at_points(candidate_points)
print("查询点的风速矢量:")
print(wind_vectors_at_points)
```

## 为路径规划算法提供信息

这个 get_wind_at_points 函数是连接物理世界和决策大脑的桥梁。不同的路径规划算法可以按以下方式利用它：

- **A* 算法**：风场信息被整合到代价函数$g(n)$（从起点到当前节点的实际代价）中。当算法探索从节点 A 移动到节点 B 时，它会查询路径段上的风矢量。如果移动方向与风向相反（逆风），则该路径段的代价值（通常代表能量消耗）会显著增加；如果顺风，则代价可以适当减少[51]。无人机要维持一定的对地速度，其所需的对空速度和推力直接受风的影响，这构成了代价计算的物理基础。
- **强化学习（RL）**：在每个时间步，无人机所在位置的局部风矢量成为智能体（Agent）观测到的状态（State）的一部分。奖励函数（Reward Function）可以被设计为惩罚高能量消耗。能量消耗可以通过无人机模型计算得出，即为了抵消风力并达到期望速度所需的推力大小[55]。
- **模型预测控制（MPC）**：风被视为一个可测量的扰动。MPC 控制器内部的动态模型会包含风力项。在每个控制周期，控制器会利用对未来风况的（短期）预测，在一个有限的时间域内优化一系列控制输入（如推力和姿态），以最小化一个包含路径跟踪误差和能量消耗的代价函数[58]。

通过将静态的 VTK 数据文件抽象为一个动态的“风场即服务”（Wind Field as a Service），路径规划模块得以在复杂的城市峡谷风场中进行智能决策，生成真正意义上的“避风”安全轨迹。

# 第五部分：执行任务——从路径到 PX4 仿真（步骤 4 → 5）

此环节的目标是将路径规划模块生成的安全轨迹（设想为 mission_path.csv），传递给 PX4 飞行仿真器执行。这一传递过程在技术上是可行的，但原始设想中采用.csv 文件作为直接输入的机制存在误解。在 PX4 生态系统中，标准且可靠的任务下发方式是通过 MAVLink 协议进行程序化上传。

## 修正关于任务传递的误解

PX4 自动驾驶仪固件本身并不具备在飞行任务中直接解析通用 CSV 文件来作为航点序列的功能[62]。尽管 CSV 格式在日志记录、数据分析或某些地面站软件的辅助功能中有所应用，但它并非 PX4 任务执行引擎的标准输入格式。

与 PX4（无论是真实硬件还是软件在环仿真 SITL）进行通信和控制的标准协议是 **MAVLink**。一个飞行任务（Mission）被定义为一系列 MISSION_ITEM 或 MISSION_ITEM_INT 消息的有序集合，并通过 MAVLink 协议从地面站（或控制计算机）上传到飞行控制器中[64]。

## 工具链深度解析：MAVSDK-Python 工作流

MAVSDK 是官方推荐的、用于与 MAVLink 系统（如 PX4）进行交互的高级软件开发工具包（SDK）。它为多种语言提供了简洁强大的 API，其中 MAVSDK-Python 是本项目实现任务自动上传的理想选择[65]。

以下是利用 MAVSDK-Python 构建自动化任务上传脚本的详细工作流程：

1. **读取 CSV 轨迹文件**：使用 Python 的 pandas 或 csv 标准库，读取路径规划模块生成的 mission_path.csv 文件，将其中的航点坐标（经度、纬度、相对高度）加载到内存中，例如一个列表或 Numpy 数组。
2. **连接到 PX4 仿真器**：
    - 启动 PX4 SITL 和 Gazebo 仿真环境。PX4 SITL 会通过 UDP 端口（默认为 14540）暴露其 MAVLink 接口。

- 在 Python 脚本中，实例化 mavsdk.System() 对象，并使用 await drone.connect(system_address="udp://:14540")来建立与仿真无人机的连接[67]。

3. **构建任务项列表（MissionItem List）**：
   - 遍历从 CSV 文件中读取的航点数据。
   - 为每个航点创建一个 mavsdk.MissionItem 对象。在创建对象时，填充其属性，包括 latitude_deg（纬度）、longitude_deg（经度）、relative_altitude_m（相对高度）、speed_m_s（飞行速度）、is_fly_through（是否悬停）等[68]。
4. **创建任务计划（MissionPlan）**：将包含所有 MissionItem 对象的列表封装到一个 mavsdk.MissionPlan 对象中。
5. **上传任务**：调用 await drone.mission.upload_mission(mission_plan)函数。此函数会将整个任务计划通过 MAVLink 协议发送给无人机。脚本应检查返回值，确保任务上传成功[69]。
6. **执行任务**：任务上传成功后，脚本可以通过 MAVSDK 发送指令来控制无人机。
   - **解锁**：await drone.action.arm()
   - **开始任务**：await drone.mission.start_mission()
   - **监控进度**（可选）：可以订阅 drone.mission.mission_progress()来异步接收任务执行的进度更新。

以下是一个完整的 mission_upload.py 脚本示例，它清晰地展示了上述流程：

Python

```
import asyncio
import csv
from mavsdk import System
from mavsdk.mission import MissionItem, MissionPlan

async def run(csv_filepath):
    """
    读取 CSV 航点文件，创建并上传任务，然后执行任务。
    """
    drone = System()
    await drone.connect(system_address="udp://:14540")

    print("等待无人机连接...")
    async for state in drone.core.connection_state():
        if state.is_connected:
            print("无人机已连接！")
            break
```

```
    # --- 从 CSV 文件读取航点 ---
    mission_items =
    with open(csv_filepath, 'r') as file:
        reader = csv.reader(file)
        next(reader) # 跳过表头
        for row in reader:
            # 假设 CSV 格式为: lat, lon, alt_m, speed_ms, fly_through
            lat, lon, alt, speed, fly_through = row
            mission_items.append(MissionItem(float(lat),
                                             float(lon),
                                             float(alt),
                                             float(speed),

is_fly_through=bool(int(fly_through)),

camera_action=MissionItem.CameraAction.NONE))

    if not mission_items:
        print("CSV 文件中未找到航点，退出。")
        return

    mission_plan = MissionPlan(mission_items)

    # --- 上传并执行任务 ---
    print("正在清空之前的任务...")
    await drone.mission.clear_mission()

    print("正在上传新任务...")
    await drone.mission.upload_mission(mission_plan)
    print("任务上传成功！")

    print("等待无人机获取 GPS 定位...")
    async for health in drone.telemetry.health():
        if health.is_global_position_ok and health.is_home_position_ok:
            print("GPS 定位已就绪。")
            break

    print("-- 解锁无人机 --")
    await drone.action.arm()

    print("-- 开始执行任务 --")
    await drone.mission.start_mission()
```

```
    # 监控任务进度
    async for mission_progress in drone.mission.mission_progress():
        print(f"任务进度: {mission_progress.current}/{mission_progress.total}")
        if mission_progress.current == mission_progress.total:
            print("任务完成！")
            break

if __name__ == "__main__":
    # 假设 CSV 文件名为 'mission_path.csv'
    asyncio.run(run('mission_path.csv'))
```

这个 Python 脚本实际上扮演了一个**无头地面控制站（Headless GCS）**的角色。它以编程方式完成了通常由 QGroundControl 或 Mission Planner 等图形化软件手动执行的操作——定义航点、上传任务、命令起飞[70]。理解这一点，就能明白为什么此环节的核心是通信协议和 SDK，而非简单的文件读写。

# 第六部分：重建数字孪生——从仿真数据到可视化（步骤 1、3、5 → 6）

这是项目的收官环节，旨在将来自不同源头的数据——城市建筑模型、CFD 风场数据和 PX4 飞行日志——整合到一个统一的可视化平台中，以“数字孪生”的形式重现整个试飞过程。这一过程在技术上是完全可行的，其成功的关键在于选择一个能够高效融合几何模型、科学数据和时序数据的现代化实时 3D 平台。

## 可行性评估：数据融合与渲染平台的选择

“AIGC 可视化”这一提法，更准确的理解应是利用具备先进渲染能力（通常由 AI 技术如光线追踪加速）的平台，对多源仿真数据进行**综合（Synthesis）与重建（Reconstruction）**，而非利用生成式 AI 凭空创造视频。其目标是创造一个高保真、物理准确的虚拟副本，忠实地再现仿真结果。

## 工具链深度解析：数据提取与可视化平台

1. **从 PX4 日志中提取真实飞行轨迹**：
    - **数据源**：PX4 仿真结束后会生成一个二进制格式的飞行日志文件，后缀为.ulg [72]。该文件详细记录了飞行过程中的所有传感器数据、内部状态估计和控制器输出。
    - **核心工具**：pyulog 是官方推荐的 Python 库，用于解析.ulg 文件 [73]。
    - **代码逻辑**：需要编写一个 Python 脚本，使用 pyulog 库打开.ulg 文件，然后：
        1. 遍历日志中的数据集（ulog.data_list）。
        2. 找到名为 vehicle_local_position 和 vehicle_attitude 的数据集。
        3. 从这些数据集中提取时间戳（timestamp）、位置（x, y, z）和姿态（通常是四元数 q）的序列。
        4. 将提取出的时序数据保存为一种通用格式，如 CSV，以便后续导入到可视化平台中。

    以下是实现该功能的 Python 脚本示例：

```
Python
import pandas as pd
from pyulog import ULog

def extract_trajectory_from_ulog(ulog_filepath, output_csv_path):
    """
    从.ulg 日志文件中提取无人机的位置和姿态数据，并保存为 CSV 文件。
    """
    ulog = ULog(ulog_filepath)

    # 提取位置数据
    try:
        pos_data = ulog.get_dataset('vehicle_local_position').data
        pos_df = pd.DataFrame(pos_data)
    except (KeyError, IndexError):
        print("警告：日志中未找到 'vehicle_local_position' 数据。")
        return

    # 提取姿态数据（四元数）
    try:
        att_data = ulog.get_dataset('vehicle_attitude').data
        att_df = pd.DataFrame(att_data)
    except (KeyError, IndexError):
        print("警告：日志中未找到 'vehicle_attitude' 数据。")
        att_df = None

    # 合并数据
    # 使用 pandas 的 merge_asof 进行时间戳对齐
    if att_df is not None:
        trajectory_df = pd.merge_asof(
            pos_df[['timestamp', 'x', 'y', 'z']],
            att_df[['timestamp', 'q', 'q', 'q', 'q']],
            on='timestamp',
            direction='nearest'
        )
    else:
        trajectory_df = pos_df[['timestamp', 'x', 'y', 'z']]

    trajectory_df.to_csv(output_csv_path, index=False)
    print(f"轨迹数据已成功提取并保存至 {output_csv_path}")

# --- 使用示例 ---
extract_trajectory_from_ulog('flight.ulg', 'trajectory.csv')
```

2. **选择并使用可视化平台**：
   - **首选推荐：NVIDIA Omniverse**：Omniverse 是专为构建物理精确的数字孪生而设计的平台。
     - **核心优势**：它以通用场景描述（USD）格式为核心，擅长整合来自不同软件（如 CAD、DCC、仿真工具）的数据。Omniverse 提供了专门用于科学数据可视化的扩展

（Extensions）。例如，omni.flowusd 扩展能够渲染体积数据，支持从.vtk 文件转换而来的 OpenVDB（.vdb）格式，实现粒子流或流线可视化[75]。

- **工作流程**:
  1. 将城市模型（.fbx）导入 Omniverse 场景。
  2. 将 CFD 输出的.vtk 风场文件转换为 OpenVDB（.vdb）格式（可使用 Houdini 或 Python 脚本完成），然后在 Omniverse 中通过 omni.flowusd 进行渲染，显示为动态的流线或粒子。
  3. 将上一步从.ulg 日志中提取的轨迹 CSV 文件导入，创建一条动画路径，并让一个无人机模型沿此路径运动。
  4. 利用 Omniverse 的 RTX 实时光线追踪渲染器，生成具有高度真实感光影效果的最终画面。

- **备选方案：Unreal Engine (UE)**：作为顶级的游戏引擎，UE 拥有无与伦比的实时渲染质量。
  - **挑战**：UE 本身不原生支持 VTK 等科学数据格式。要导入风场数据，需要依赖第三方插件（如[86]中提到的 VTKPlugin，需要自行编译 VTK）或开发自定义的 C++模块来解析.vtk 文件，并将其数据转化为 UE 内部的粒子系统或体积纹理[77]。这是一个巨大的开发工作量。
- **备选方案：Unity**：与 UE 类似，Unity 也是一个强大的游戏引擎，但同样缺乏对科学数据格式的内建支持。
  - **挑战**：集成 VTK 数据通常需要借助商业插件，如 VTKUnity-Activiz，但该插件价格昂贵且可能存在平台限制[80]。开源方案存在，但配置复杂[84]。Unity 内置的 WindZone 组件仅用于视觉特效，无法用于显示真实的 CFD 数据[85]。

综上所述，NVIDIA Omniverse 凭借其为数字孪生和科学可视化量身打造的生态系统，是完成此最终集成步骤的最直接、最高效的平台。它将几何模型、动态流场和精确的飞行轨迹这三个核心元素无缝地融合在一个场景中，最终呈现出一个可交互、高保真、物理准确的数字孪生试飞环境。

# 第七部分：综合评估与战略建议

通过对整个研究项目流程的逐层剖析，可以得出结论：该设想是一个宏大且具有挑战性的系统集成工程。其成功不仅取决于各个独立模块的执行质量，更关键的是在于如何构建稳定、高效的数据桥梁来连接这些异构系统。本节将对整个工作流进行综合，提供最终的技术栈建议，并识别关键风险及其规避策略。

## 端到端验证工作流

下图修正并细化了原始的“纸上谈兵”流程，展示了一个经过可行性验证的、实际可操作的技术路线图。图中明确了每个步骤所使用的核心软件/库，以及每次数据传递所依赖的关键文件格式和处理方法。

Code snippet

```
graph TD
    subgraph "步骤 1: 世界构建"
        A1 --> B(ArcGIS / Blender-OSM);
        A2["AIGC 生成 (港城)"] --> B;
        B -- "输出视觉模型" --> C[".obj /.fbx"];
    end

    subgraph "步骤 2: 物理预测-气象"
        D -- "输出气象场" --> E["wrfout_*.nc"];
    end

    subgraph "步骤 3: 物理预测-CFD"
        F(ANSYS SpaceClaim);
        G(ANSYS Fluent);
        C -- "1. 导入" --> F;
        F -- "2. 几何修复、简化、构建流体域" --> H;
```

```
        H -- "3. 网格划分" --> G;
        E -- "4. Python 脚本 (xarray/wrf-python) 解析" --> I[".prof 边界文件"];
        I -- "5. 导入边界条件" --> G;
        G -- "6. CFD 计算" --> J["city_windfield.vtk"];
    end

    subgraph "步骤 4: 路径规划"
        K(路径规划算法);
        J -- "PyVista 库加载" --> L["风场查询 API (Python)"];
        L -- "实时查询风矢量" --> K;
        K -- "生成安全轨迹" --> M["mission_path.csv"];
    end

    subgraph "步骤 5: PX4 仿真"
        N(PX4 SITL + Gazebo);
        M -- "MAVSDK-Python 脚本读取" --> O["生成 MAVLink 任务"];
        O -- "通过 UDP 上传" --> N;
        N -- "执行仿真并记录" --> P["flight_log.ulg"];
    end

    subgraph "步骤 6: 可视化重建"
        Q(NVIDIA Omniverse);
        C -- "导入建筑模型" --> Q;
        J -- "转换为.vdb 后导入流场" --> Q;
        P -- "pyulog 脚本解析" --> R["轨迹数据.csv"];
        R -- "导入飞行路径" --> Q;
    end

    Q -- "渲染最终数字孪生画面" --> S[数字孪生试飞画面];
```

## 整合技术栈与技能要求

为了成功实施上述工作流，项目团队需要掌握以下软件工具和编程技能。

**表 2：所需软件与技能**

| 类别 | 软件/库 | 主要用途 | 所需核心技能 |
| --- | --- | --- | --- |
| **世界构建** | ArcGIS / CityEngine / Blender (含 Blosm 插件) | 生成三维城市模型 | GIS 数据处理、程序化建模、3D 建模 |
| **几何处理** | ANSYS SpaceClaim | 修复、简化模型，构建 CFD 流体域 | 专业级 CAD 操作、几何拓扑理解 |
| **气象模拟** | WRF (或同类中尺度模型) | 生成大尺度气象背景场 | 气象学、数值天气预报模型操作 |
| **CFD 仿真** | ANSYS Fluent | 计算微尺度城市风场 | 计算流体动力学、网格划分、边界条件设置 |
| **飞控仿真** | PX4 SITL / Gazebo | 模拟无人机飞行物理与控制逻辑 | 机器人操作系统(ROS)、无人机飞控原理 |
| **可视化** | NVIDIA Omniverse | 集成多源数据，进行高保真实时渲染 | 实时 3D 引擎操作、场景搭建、USD 工作流 |
| **核心编程** | Python 3.8+ | 编写所有数据传递和自动化脚本 | 精通 Python 编程 |
|  | └ pyproj | 统一坐标系转换 | 地理空间数据处理 |
|  | └ xarray，netCDF4，wrf-python | 解析 WRF 气象数据 | 科学数据处理、Numpy |

| | | | |
|---|---|---|---|
| | └ pyvista | 解析 VTK 风场数据，提供查询 API | 科学可视化、Numpy |
| | └ mavsdk | 与 PX4 通信，上传和执行任务 | 异步编程 (asyncio)、网络编程 |
| | └ pyulog, pandas | 解析 PX4 飞行日志 | 数据分析、Numpy |

## 关键项目风险与规避策略

1. **风险一：几何处理工作量严重超估**
   - **描述**：从视觉模型到 CFD 分析模型的转换（步骤 1→3）是整个项目中技术不确定性最高、最耗费人力的环节。其所需时间可能远超预期，导致项目延期。
   - **规避策略**：
     - **分阶段实施**：不要一开始就处理整个广州市模型。首先选取一个代表性的小区域（如几个街区）作为试点，完整地走通几何处理流程，以此为基准来科学评估处理整个模型所需的时间和资源。
     - **简化优先**：在满足研究目标的前提下，尽可能地简化模型。过度追求几何细节会使工作量呈指数级增长。
2. **风险二：定制化脚本的复杂性与健壮性**
   - **描述**：项目依赖至少三个核心的定制 Python 脚本（WRF-to-Fluent, VTK-Query-API, CSV-to-MAVSDK）。这些脚本中的任何一个出现错误或效率低下，都会导致整个流程中断或结果失真。
   - **规避策略**：
     - **模块化开发**：将每个脚本视为一个独立的软件开发任务，为其编写清晰的文档、接口定义和单元测试，确保其在各种输入下的稳定性和正确性。
     - **技能储备**：确保团队中拥有精通 Python 科学计算栈（Numpy, Scipy, Pandas）和异步编程（asyncio）的成员。
3. **风险三：坐标系统管理失误**
   - **描述**：在多环节的数据传递中，任何一次坐标转换的失误或疏忽，都可能导致最终数据在空间上无法对齐，使整个仿真结果作废。这是一个“一招不慎，满盘皆输”的风险点。
   - **规避策略**：

- **前置策略**：严格执行第一部分中提出的“统一坐标系”策略，将其作为项目启动的第一个技术任务。
- **集中管理**：将所有坐标转换逻辑集中到一个独立的、经过充分测试的 Python 模块中，项目其他所有部分都调用此模块进行转换，避免在各处重复实现，从而减少出错可能。

## 最终建议

该研究项目是一个富有远见且技术上可行的系统集成挑战。其核心并非在于单个软件的深度使用，而在于如何利用编程技术和系统思维，将这些强大的“孤岛”工具链有效地连接起来。

建议采用**迭代式、增量化**的开发方法。首先，使用一个极度简化的几何模型（例如，几个立方体代表建筑）和粗糙的仿真参数，快速打通从头到尾的整个数据链路。在确保整个流程可以自动化运行后，再逐步替换为更精细的城市模型和更高分辨率的物理仿真，不断提升数字孪生的保真度。这种方法有助于及早暴露集成问题，降低项目风险，确保最终能够成功构建一个功能完备、物理可信的城市无人机数字孪生仿真平台。

## Works cited


1. Geographic datum transformations—ArcGIS Pro | Documentation, accessed October 17, 2025, https://pro.arcgis.com/en/pro-app/latest/help/mapping/properties/geographic-coordinate-system-transformation.htm
2. Coordinate systems, map projections, and transformations—ArcGIS Pro | Documentation, accessed October 17, 2025, https://pro.arcgis.com/en/pro-app/latest/help/mapping/properties/coordinate-systems-and-projections.htm
3. Local tangent plane coordinates - Wikipedia, accessed October 17, 2025, https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates
4. North East Down (NED) gazebo frame - Gazebo Answers archive, accessed October 17, 2025, http://answers.gazebosim.org/question/12396/
5. Proj - pyproj 3.7.2 documentation - GitHub Pages, accessed October 17, 2025, https://pyproj4.github.io/pyproj/stable/api/proj.html
6. Getting Started - pyproj 3.7.2 documentation - GitHub Pages, accessed October 17, 2025, https://pyproj4.github.io/pyproj/stable/examples.html
7. Importing cad model to Ansys fluent : r/CFD - Reddit, accessed October 17, 2025, https://www.reddit.com/r/CFD/comments/1avpswl/importing_cad_model_to_ansys_fluent/
8. How to import external 3D geometry in Ansys fluent | Solidworks 3D | English - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=L4r2zIfyNRA
9. HOW TO IMPORT MODEL IN ANSYS FLUENT - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=PWKJQKSabBU
10. Generating 3D Models for CFD | Resolved Analytics, accessed October 17, 2025, https://www.resolvedanalytics.com/cfd-in-practice/how-to-create-3d-geometry-for-cfd
11. Preparing the Geometry for CFD Process using Rhino - Cloud Towing Tank, accessed October 17, 2025, https://cloudtowingtank.com/preparing-the-geometry-for-cfd-process-using-rhino/
12. 5 Steps to prepare CAD for CFD Simulation…The Easy Way! - Simcenter, accessed October 17, 2025, https://blogs.sw.siemens.com/simcenter/5_steps_for_cfd_cad_preparation/
13. Need Efficient Cad cleanup tips in Ansys Spaceclaim : r/CFD - Reddit, accessed October 17, 2025, https://www.reddit.com/r/CFD/comments/1hwlit9/need_efficient_cad_cleanup_tips_in_ansys/
14. Tutorial 5: Import initial shapes—ArcGIS CityEngine Resources | Documentation, accessed October 17, 2025,

https://doc.arcgis.com/en/cityengine/2024.1/tutorials/tutorial-5-import-initial-shapes.htm
15. Blosm for Blender: OpenStreetMap, Google 3D cities, terrain - GitHub, accessed October 17, 2025, https://github.com/vvoovv/blosm
16. AI-powered automated model construction for patient-specific CFD simulations of aortic flows - PMC, accessed October 17, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC12412661/
17. Conceptual design using generative AI and CFD simulations on AWS | AWS HPC Blog, accessed October 17, 2025, https://aws.amazon.com/blogs/hpc/conceptual-design-using-generative-ai-and-cfd-simulations-on-aws/
18. ANSYS Complete Course | 07. Spaceclaim Geometry Cleanup - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=w0_MrpU1rdA
19. Defeature Geometry for Simulation in 3 min with ANSYS SpaceClaim - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=n1361hvmtas
20. SpaceClaim Tutorial: A Beginners Guide to SpaceClaim DM - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=lt_GUsxJXCM
21. Geometry Clean Up with SpaceClaim | Ansys Knowledge, accessed October 17, 2025, https://innovationspace.ansys.com/knowledge/forums/topic/geometry-clean-up-with-spaceclaim/
22. SpaceClaim Feature Demo - Clean Up Dirty Geometry from IGES and STEP - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=N41LZg4JxuU
23. How to Prepare CAD for CFD Simulation using Ansys Discovery - Lesson 6, accessed October 17, 2025, https://innovationspace.ansys.com/courses/index.php/courses/geometry-prep-for-cfd-using-ansys-discovery/lessons/how-to-prepare-cad-for-cfd-simulation-using-ansys-discovery-lesson-6/
24. ANSYS SpaceClaim Model Preparation for CFD - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=fgxD6pP-yrs
25. Ansys Fluent Meshing | Watertight and Fault Tolerant Workflows - SimuTech Group, accessed October 17, 2025, https://simutechgroup.com/fluent-meshing-watertight-and-fault-tolerant-workflows/
26. How to Mesh Watertight CFD Geometry in the New Fluent Task-based Workflow - Ansys, accessed October 17, 2025, https://www.ansys.com/blog/watertight-cfd-geometry-ansys-fluent
27. A Multi-Scale WRF–CFD Coupling method for High-Resolution Urban Wind Field Simulation in Shanghai Sea–Land Breeze Scenarios. | Request PDF - ResearchGate, accessed October 17, 2025, https://www.researchgate.net/publication/395008091_A_Multi-Scale_WRF-CFD_Coupling_method_for_High-

Resolution_Urban_Wind_Field_Simulation_in_Shanghai_Sea-Land_Breeze_Scenarios
28. Simulating Urban Flow and Dispersion in Beijing by Coupling a CFD Model with the WRF Model, accessed October 17, 2025, http://www.iapjournals.ac.cn/aas/article/doi/10.1007/s00376-013-2234-9
29. Coupling of WRF and building-resolving urban CFD models for analysis of strong winds over an urban area - Ams.Confex.Com., accessed October 17, 2025, https://ams.confex.com/ams/14Meso15ARAM/webprogram/Manuscript/Paper190752/ExtendedAbstract_nakayama.pdf
30. Simulating Urban Flow and Dispersion in Beijing by Coupling a CFD Model with the WRF Model - SciEngine, accessed October 17, 2025, https://www.sciengine.com/doi/10.1007/s00376-013-2234-9
31. A Study on Microscale Wind Simulations with a Coupled WRF–CFD Model in the Chongli Mountain Region of Hebei Province, China - MDPI, accessed October 17, 2025, https://www.mdpi.com/2073-4433/10/12/731
32. Tutorial on how to import, subset, aggregate and export CAMS Data - GitHub Pages, accessed October 17, 2025, https://ecmwf-projects.github.io/copernicus-training-cams/da-read-write.html
33. Python Notebook Example on how to convert a netcdf file to csv file - GitHub Gist, accessed October 17, 2025, https://gist.github.com/copernicusmarinegist/b57417225d0d4ea47c5d6200f9d8cac3
34. Reading and writing files - Xarray documentation, accessed October 17, 2025, https://docs.xarray.dev/en/stable/user-guide/io.html
35. Is it possible to write a .csv file from a xarray.Dataset in python? - Stack Overflow, accessed October 17, 2025, https://stackoverflow.com/questions/71665819/is-it-possible-to-write-a-csv-file-from-a-xarray-dataset-in-python
36. How To Use — wrf-python 1.4.0 documentation, accessed October 17, 2025, https://wrf-python.readthedocs.io/en/latest/basic_usage.html
37. NCAR/wrf-python: A collection of diagnostic and interpolation routines for use with output from the Weather Research and Forecasting (WRF-ARW) Model. - GitHub, accessed October 17, 2025, https://github.com/NCAR/wrf-python
38. ANSYS Fluent Fully Developed Flow UDF : r/CFD - Reddit, accessed October 17, 2025, https://www.reddit.com/r/CFD/comments/1jep1bd/ansys_fluent_fully_developed_flow_udf/
39. How create and use Profile txt file as a boundary condition in FLUENT?, accessed October 17, 2025, https://innovationspace.ansys.com/forum/forums/topic/how-create-and-use-profile-txt-file-as-a-boundary-condition-in-fluent/
40. ANSYS Fluent: How to Write/Read Boundary Profiles | Tips & Tricks - YouTube, accessed October 17, 2025, https://www.youtube.com/watch?v=-42b0tZufP8
41. How do you input cell zone conditions in ANSYS Fluent? - ResearchGate, accessed

October 17, 2025, https://www.researchgate.net/post/How-do-you-input-cell-zone-conditions-in-ANSYS-Fluent
42. ANSYS FLUENT 12.0 User's Guide - 7.6.2 Profile File Format, accessed October 17, 2025, https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node265.htm
43. ANSYS FLUENT 12.0 User's Guide - 7.6.3 Using Profiles, accessed October 17, 2025, https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node266.htm
44. PyVista & VTK, accessed October 17, 2025, https://tutorial.pyvista.org/tutorial/06_vtk/index.html
45. Basic API Usage - Wrapping a VTK Data Object - PyVista, accessed October 17, 2025, https://docs.pyvista.org/user-guide/simple.html
46. pyvista.read, accessed October 17, 2025, https://docs.pyvista.org/api/utilities/_autosummary/pyvista.read.html
47. Compare interpolation/sampling methods — PyVista 0.46.3 ..., accessed October 17, 2025, https://docs.pyvista.org/examples/01-filter/interpolate_sample.html
48. pyvista.DataSetFilters.interpolate, accessed October 17, 2025, https://docs.pyvista.org/api/core/_autosummary/pyvista.datasetfilters.interpolate
49. Detailed Interpolating Points — PyVista 0.46.3 documentation, accessed October 17, 2025, https://docs.pyvista.org/examples/01-filter/interpolate.html
50. Cost Optimization of Wind Turbines for Large-Scale Offshore Wind Farms - OSTI, accessed October 17, 2025, https://www.osti.gov/etdeweb/servlets/purl/605630
51. Cost optimization of wind turbines for large-scale offshore wind farms - DTU Research Database, accessed October 17, 2025, https://orbit.dtu.dk/files/7731717/ris_r1000.pdf
52. 3d-pathfinding · GitHub Topics, accessed October 17, 2025, https://github.com/topics/3d-pathfinding
53. rhaeus/path_planning_3d: Read a map file, create a 3D grid, plan a 3D path using A, accessed October 17, 2025, https://github.com/rhaeus/path_planning_3d
54. Energy-aware Goal Selection and Path Planning of UAV Systems via Reinforcement Learning, accessed October 17, 2025, https://swapp.cs.iastate.edu/files/inline-files/niaraki_ea_arxiv-march2020.pdf
55. Reinforcement Learning-Based Energy-Saving Path Planning for UAVs in Turbulent Wind, accessed October 17, 2025, https://www.mdpi.com/2079-9292/13/16/3190
56. Dynamic Path Planning with Reinforcement Learning - Anvil Labs, accessed October 17, 2025, https://anvil.so/post/dynamic-path-planning-with-reinforcement-learning
57. Investigation of Wind Effects on UAV Adaptive PID Based MPC Control System - Redalyc, accessed October 17, 2025, https://www.redalyc.org/journal/5722/572277355009/html/
58. Investigation of Wind Effects on UAV Adaptive PID Based MPC Control System, accessed October 17, 2025, https://www.researchgate.net/publication/379491290_Investigation_of_Wind_Effects_

on_UAV_Adaptive_PID_Based_MPC_Control_System
59. Towards Model Predictive Control for Acrobatic Quadrotor Flights - arXiv, accessed October 17, 2025, https://arxiv.org/html/2401.17418v1
60. Predictive Control of a Wind Turbine Based on Neural Network-Based Wind Speed Estimation - MDPI, accessed October 17, 2025, https://www.mdpi.com/2071-1050/15/12/9697
61. Where can I store and read from a .csv file? - PX4 Autopilot, accessed October 17, 2025, https://discuss.px4.io/t/where-can-i-store-and-read-from-a-csv-file/2268
62. How to read from .csv file into controller module? - PX4 Discussion Forum, accessed October 17, 2025, https://discuss.px4.io/t/how-to-read-from-csv-file-into-controller-module/2525
63. Mission Protocol - MAVLink Guide, accessed October 17, 2025, https://mavlink.io/en/services/mission.html
64. MAVSDK (main / v3) | MAVSDK Guide, accessed October 17, 2025, https://mavsdk.mavlink.io/
65. MAVSDK-Python, accessed October 17, 2025, https://mavsdk.mavlink.io/main/en/python/
66. Python QuickStart | MAVSDK Guide, accessed October 17, 2025, https://mavsdk.mavlink.io/main/en/python/quickstart.html
67. Missions | MAVSDK Guide, accessed October 17, 2025, https://mavsdk.mavlink.io/main/en/cpp/guide/missions.html
68. Mission — MAVSDK-Python 3.10.2 documentation, accessed October 17, 2025, http://mavsdk-python-docs.s3-website.eu-central-1.amazonaws.com/plugins/mission.html
69. Copter SITL/MAVProxy Tutorial — Dev documentation - ArduPilot, accessed October 17, 2025, https://ardupilot.org/dev/docs/copter-sitl-mavproxy-tutorial.html
70. MAVProxy documentation - ArduPilot, accessed October 17, 2025, https://ardupilot.org/mavproxy/
71. ULog File Format | PX4 Guide (main), accessed October 17, 2025, https://docs.px4.io/main/en/dev_log/ulog_file_format.html
72. PX4/pyulog: Python module & scripts for ULog files - GitHub, accessed October 17, 2025, https://github.com/PX4/pyulog
73. Flight Log Analysis | PX4 Guide (main), accessed October 17, 2025, https://docs.px4.io/main/en/log/flight_log_analysis
74. Overview — omni.kit.cadence_reality_dc_design 2.0.1 documentation, accessed October 17, 2025, https://docs.omniverse.nvidia.com/extensions/latest/ext_cadence_reality_dc_design.html
75. Using Omniverse™ Flow, accessed October 17, 2025,

https://docs.omniverse.nvidia.com/extensions/latest/ext_fluid-dynamics/using.html
76. Game Engines for Immersive Visualization: Using Unreal Engine Beyond Entertainment, accessed October 17, 2025, https://arxiv.org/html/2411.02090v1
77. [Paraview-developers] What I have learned about providing a VR system that is intuitive to use while keeping discomfort to a minimum., accessed October 17, 2025, https://www.paraview.org/pipermail/paraview-developers/2016-November/004833.html
78. VTK Array doesn't work correctly if it's used under Unreal Engine, accessed October 17, 2025, https://discourse.vtk.org/t/vtk-array-doesnt-work-correctly-if-its-used-under-unreal-engine/15812
79. How to integrate VTK with Unity - Development, accessed October 17, 2025, https://discourse.vtk.org/t/how-to-integrate-vtk-with-unity/1108
80. Using VTK with unity for reading .VTU files but not Visualizing. Mac OS, accessed October 17, 2025, https://discourse.vtk.org/t/using-vtk-with-unity-for-reading-vtu-files-but-not-visualizing-mac-os/7894
81. VTKUnity-Activiz | Tutorial Projects | Unity Asset Store, accessed October 17, 2025, https://assetstore.unity.com/packages/essentials/tutorial-projects/vtkunity-activiz-163686
82. Transfering Paraview to Unity, accessed October 17, 2025, https://discourse.paraview.org/t/transfering-paraview-to-unity/13962
83. EnlitHamster/VtkUnityWorkbench: VTK Unity plugin enabling visualization creation in VR, accessed October 17, 2025, https://github.com/EnlitHamster/VtkUnityWorkbench
84. Wind Zones - Unity - Manual, accessed October 17, 2025, https://docs.unity3d.com/2020.1/Documentation/Manual/class-WindZone.html
85. VRGroupRWTH/VTKPlugin: A plugin that downloads ... - GitHub, accessed October 17, 2025, https://github.com/VRGroupRWTH/VTKPlugin