Technical Report: Evaluation and Selection of Simulators for the SkyVision Multi-Drone Project

Ahmed, Belal, Abdllah, Ammar, Yousef
Al Alamein International University
New Alamein City, Egypt

***Abstract*—This technical report documents our evaluation process and findings regarding several open-source simulators and middleware for the SkyVision multi-drone project. We systematically explored Gazebo, AirSim, Isaac, as our simulators and PX4, QGroundControl, and ROS2, comparing their features, integration complexity, sensor and physics realism, and suitability for collaborative mapping and control tasks specific to SkyVision. The report details our hands-on experiments, highlights the strengths and limitations of each platform, and explains our decision to adopt AirSim with ROS2 as the foundation for SkyVision's simulation and development work. Our experience aims to guide other drone researchers and students in selecting appropriate simulation tools for aerial robotics projects.**

I. Introduction

SkyVision is a specialized graduation project in collboration with JMU focused on developing an autonomous swarm of Unmanned Aerial Vehicles (UAVs) designed for rapid disaster response. The system leverages Artificial Intelligence to coordinate multiple drones for search-and-rescue operations, utilizing real-time Computer Vision (CV) and Simultaneous Localization and Mapping (SLAM).

This report documents the selection, configuration, and evaluation of the simulation environment used to develop and test SkyVision before physical deployment.

II. Project Requirements

To successfully simulate the SkyVision swarm, the simulation platform needed to meet specific criteria:

Multi-Vehicle Support: Capability to spawn and control multiple drones simultaneously (Swarm functionality).

ROS2 Integration: Native compatibility with ROS2 (Humble/Jazzy) for communication between agents we are currently using humble for better compatibility.

Sensor Simulation: High-fidelity simulation of RGB cameras, Lidar, and IMUs for Computer Vision and SLAM tasks.

Physics Fidelity: Accurate flight dynamics compatible with the PX4 Autopilot stack.

Customizability: Ability to modify drone models (e.g., Holybro x500) and environments (disaster scenarios).

III. Simulator and Middleware Options

We evaluated three primary candidates for the SkyVision simulators:

AirSim A simulator built on Unreal Engine by Microsoft. It offers photorealistic rendering and APIs for controlling drones.

Pros: Extremely high visual fidelity (great for CV).

Cons: Deprecated/Archived by Microsoft; heavy resource usage; difficult setup with modern ROS2 versions.

there are some community maintained forks that are compatible with newer versions of unreal and ROS2.

Gazebo (Harmonic) The open-source standard for robotics simulation, previously known as Ignition. It is the default simulator for ROS2.

Pros: Native ROS2 integration; lightweight; official support from PX4 Autopilot; huge community.

Cons: Visuals are less "game-like" than Unreal Engine (though Harmonic is significantly improved).

NVIDIA Isaac Sim A scalable robotics simulation application based on NVIDIA Omniverse.

Pros: Photorealistic physics and visuals; GPU-accelerated.

Cons: Requires specific high-end NVIDIA hardware; steeper learning curve for custom drone integration compared to Gazebo.

IV. Evaluation Process

We compared the simulators based on the following metrics:

Integration Ease: How easily does it connect with PX4 and ROS2?

Swarm Scalability: Can it run 3+ drones without crashing the host PC?

Sensor Data: Ease of accessing camera/Lidar streams in Python/OpenCV.

Long-term Viability: Is the software actively maintained?

V. Setup and Experiments

Installation and Configuration Middleware: We installed ROS2 Humble on Ubuntu 22.04.

Simulator: We set up Gazebo Harmonic using the official OSRF repositories.

Autopilot: We cloned the PX4-Autopilot repository and configured the build toolchain. Integration Steps SITL Bridging: Configured the MicroXRCE-DDS Agent to bridge topics between PX4 (running in simulation) and ROS2 nodes.

Model Creation: Created a custom SDF (Simulation Description Format) file for the Holybro x500 frame, attaching virtual sensors (2D Lidar and RGB Camera).

Swarm Launching: Wrote a shell script to spawn multiple instances of px4_sitl with unique namespace IDs (e.g., drone1, drone2) and distinct UDP ports for MAVLink communication.

VI. Comparison and Decision