Ministry of Education and Science of Ukraine

National University of "Kyiv-Mohyla Academy"

Department of Informatics of the Faculty of Informatics

Comparative Analysis of Development Environments for UAV Software Development

Text part to the thesis in the specialty "Computer Sciences" - 122

Thesis Supervisor:

Senior Lecturer

Andrew Kurochkin

_____ (Signature)

"___" _____ 2025

Done by Student:

Sofiia Budilova

"___" _____ 2025

Kyiv, 2025

Ministry of Education and Science of Ukraine
National University of "Kyiv-Mohyla Academy"
Department of Informatics of the Faculty of Informatics

Approved
Head of Department of Informatics,
Associate Professor, Ph.D.
S. S. Gorokhovskyi

_____ (Signature)

"___" _____ 2025

INDIVIDUAL TASK for thesis
of the student Sofiia Budilova
4th year of the Faculty of Informatics
topic: Comparative Analysis of Development Environments for UAV Software Development

The content of the PM to the thesis:
Abstract
Introduction
Overview and Evaluation of Related Papers
UAV Simulators Overview
Analysis of the Distribution of Development Environments and Flight Software
Identifying the Research Gap
Development of a UAV Testing Environment with a Moving Object
Conclusion and Future Work
Bibliography

Date of issue: "___" _____ 2025

Supervisor: _____ (Signature)

Task received: _____ (Signature)

Abstract

Unmanned aerial vehicles are in demand nowadays due to their ability to perform various tasks, both military and civilian, without the involvement of humans. The UAV simulators (UAV algorithm development environments) have a great value at the present day since they present a way to test all the new UAV algorithms that are spreading more and more finding real-world applications across various disciplines.

A plethora of simulators already exist. All of them have their own advantages and disadvantages. This presents difficulties for developers to opt for the most suitable one to meet their requirements. This paper reviews the most popular UAV simulators. It also provides statistics regarding the amount of papers in general and throughout the years related to simulators and also to flight control software (e.g., ArduPilot). This provides the opportunity to observe tendencies in UAV simulation technologies.

Subsequently, research was conducted to analyze scenes (usually called worlds) of the most popular and robust simulator, namely Gazebo, that are available open-source online. A list of them is provided in this thesis.

Eventually, a research gap was found, namely a shortage of Gazebo worlds containing moving objects. A new world with a moving car was created and might be used, for example, for the testing of the UAV object-tracking algorithms.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

UAV	**U**nmanned **A**erial **V**ehicle
HITL	**H**ardware **I**n **T**he **L**oop
SITL	**S**oftware **I**n **T**he **L**oop
IMU	**I**nertial **M**easurement **U**nit
LIDAR	**L**ight **I**dentification, **D**etection **a**nd **R**anging

Chapter 1

Introduction

1.1 Key definitions

In order for this thesis to be clear for those who read it, there are detailed explanations of main concepts considered in this work.

1.1.1 Unmanned aerial vehicle (UAV)

It may also be called a drone (Figure 1.1). In this thesis a drone and a UAV are used interchangeably, meaning the same thing. A drone is an aircraft with no human pilot, crew, or passengers onboard, but rather is controlled remotely or is autonomous [1, 2].

There exists a wide variety of unmanned aerial vehicles. They might be classified by their size, weight, range, speed etc. It is also worth mentioning that drones vary in their level of autonomy: while some of them require full control of a remote human operator, others have either limited autonomous abilities (e.g., a return-to-base operation) or are fully autonomous. The latter case is, however, rare and does not go beyond competitions or experimental flights [3].

Another interesting detail about unmanned aircraft systems is the input and output that the system operates with. The inputs usually include IMU (inertial measurement unit) that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers. The second type of input data is a camera input. At the same time, outputs of the system include commands to rotor thrusts (thrust control) [3].

Figure 1.1: Iris quadcopter drone [4]

1.1.2 Development environment for UAV algorithms

It may also be called a UAV simulator, flight simulation system, simulation drone platform. In this thesis the term "UAV simulator" is primarily used.

A UAV simulator is a software application that provides functionality to interact with a UAV without actually having a real physical UAV.

Flight simulation is a software that generates sensor values and the state of a flying vehicle. The physics is modeled to replicate the way a vehicle responds to flight controls and environmental conditions, e.g., weather conditions, wind, rain, air density etc. Also, the interaction with other objects and vehicles is simulated. Equations that are based on inputs from flight controls and an environment are used to update the vehicle's state.

Simulators are used for various purposes, such as pilot training, drone software development, and testing new software. Some UAV simulators are designed to generate data for training machine learning (ML) models. This thesis primarily focuses on the software testing aspect of development.

1.1.3 Flight control software

A flight control software (also known as flight control computer) receives and processes data from sensors throughout the aircraft, such as IMU, barometer, magnetometer, GPS, etc. These sensors monitor speed of the vehicle, its position and orientation in three-dimensional space. Embedded within integrated avionics packages, it executes critical functions such as guidance, navigation. A dedicated flight control computer handles high-level computational tasks, including routing, autopilot functions, and flight management. This computer interfaces with the avionics system and is responsible for displaying flight data on the cockpit's flight deck [5].

1.2 Why are UAV simulators needed?

Nowadays the UAV usage increases rapidly. UAVs are used in all kinds of surveillance, like forest fire monitoring, river monitoring, weather monitoring etc., also they are considered to be used for product deliveries. Other than that, UAVs are actively used in the military sphere, in search and rescue operations, in humanitarian aid and disaster relief.

Unmanned Aerial Vehicles (UAVs) represent one of the most relevant emerging technologies in the geoscience and remote sensing fields of the last two decades. They have become a popular instrument for a wide range of applications and replaced other platforms thanks to their flexibility and (relatively) moderate costs [6].

To accomplish named above tasks, UAVs should be able to solve the following problems:

- target detection [7],
- path planning [8],
- obstacle avoidance algorithms [9],
- payload transportation [10],
- and many more.

To enhance the performance and effectiveness, drone swarms [11] might be used. For example, multiple drones communicating with each other can scan the territory faster, which may be profitable in some cases.

Another trend that occurs for the last decades is an increase of drones' autonomy, which enlarges the number of tasks that can be performed without human intervention or control. Competitions for fully autonomous drones, which exist since 2016, put to the test our skills and knowledge about unmanned aircraft systems and all the layers that have to be autonomous. In a standard scheme, those are three layers, namely perception, planning, and control. The utilized algorithms can be either empirical or based on machine learning/deep learning. And even though improvements in speed and maneuvers are observed over the years, we are far away from having solved autonomous drone racing [3].

In order to make sure that the algorithm and the system as a whole operate properly, testing should be conducted beforehand. If the newly written algorithms were tested immediately in the real world with real hardware and vehicles, it would have done considerably more damage to the environment and to the drone itself. Such tests are more expensive and time consuming meaning they should be conducted at the last.

That is why SITL (software-in-the-loop) and HITL (hardware-in-the-loop) tests exist.

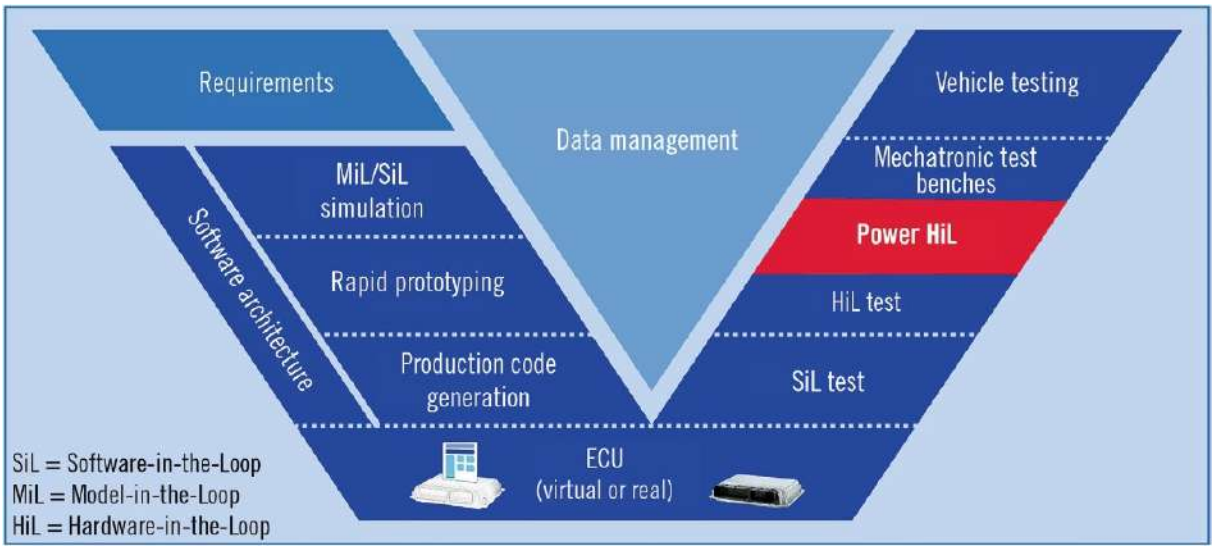

Figure 1.2: Validation procedure in the development cycle [12]

1.2.1 HITL vs SITL

SITL runs on a development computer in a simulated environment, and uses firmware specifically generated for that environment. Other than simulation drivers to provide fake environmental data from the simulator the system behaves normally.

By contrast, HITL runs normal PX4 firmware in *"HITL mode"*, on normal hardware. The simulation data enters the system at a different point than for SITL. Core modules like commander and sensors have HITL modes at startup that bypass some of the normal functionality.

In summary, HITL runs PX4 (or another software) on the actual hardware using standard firmware, but SITL actually executes more of the standard system code [13].

For now, there also occurred a new approach, an alternative to HITL, called Simulation-In-Hardware. It offers a hard real-time simulation directly on the hardware autopilot. This simulator is implemented in C++ as a PX4 module directly in the Firmware code [14].

Finally, UAV simulators are required in both SITL and HITL test stages.

1.3 Types of Development Environments for UAV Algorithms

Since there are lots of different types of UAV applications and therefore UAV algorithms, their requirements to development environments may vary vastly. For example, some may need a nice graphical picture. Others provide a wide spector of supported sensors. Some are capable of simulating various weather and environmental conditions.

1.4 Motivation

Why is one more UAV comparison needed?

Firstly, it gathers and analyzes in one place works that already exist pointing out their strong and weak sides. Readers of this thesis might find it a good starting point for becoming familiar with the UAV simulators' topic. The reviewed papers could be used for deepening the knowledge thereafter.

Secondly, new analyses in general can clarify the current state of the development of the technologies under consideration. What achievements are already done, what objectives are presented to the community, what overall trends exist – all the questions might find their answers in comparison papers from different years.

And the final purpose of this comparison is to find a problem, for which a solution could be provided within this thesis. By reviewing simulators, the conclusions about their state might reveal present weaknesses. The goal is to find such a weakness and make a move towards removing or at least mitigating it.

1.5 Thesis Structure

In Chapter 2 some of the related works were reviewed. An overview of four UAV simulators was described in Chapter 3. Statistics about UAV simulators and flight control software based on Google Scholar and Semantic Scholar was presented in Chapter 4. A research on the most popular UAV simulator Gazebo with a deeper analysis of its worlds was carried out in Chapter 5. A new Gazebo world that is designated to reduce the research gap and the solution to how this world was implemented is located in Chapter 6. Conclusions and visions of the future work are written in Chapter 7.

Chapter 2

Overview and Evaluation of Related Papers

To get acquainted with the topic of UAVs, Paper [15] might be well-suited since it explain how drone system functions, what UAV types exist, and what future tasks and challenges in this field are.

There are already papers that review UAV simulators and draw conclusions on what software suits best for what aims.

In the paper [16], published in 2018, several simulators are mentioned. Other than that, ground control stations are reviewed.

In the paper [17], published in 2021, there is a small section with comparison of a few simulators for UAVs (AirSim, Flightmare, Gazebo, Webots).

Paper [18] is also devoted to UAV simulators comparison, written in September 2024. 44 simulators were reviewed with 14 ones being reviewed in-depth. In [18], a table with criteria for simulators is given (Table 2.1). This list of criteria covers all the essential components and requirements for a UAV simulator: realistic visuals for tasks that involve processing of a camera input, integration with autopilots to test algorithms in conditions resembling real-life scenario, quality of physics engine to emulate dynamics similar to the real world, etc. The availability of a diverse range of UAV models also facilitates the development and is important during selecting the suitable software.

Paper [19] reviews flight hardware (controllers, e.g., Pixhawk), flight software, and flight simulators. These three components are closely connected in UAV software development and are essential.

Criteria	Decision Factors
Physics Fidelity	Required fidelity of physics and dynamics model for the intended use case
Visual Fidelity	Level of realism in images (e.g., for computer vision or Machine Learning (ML) applications)
Autopilots	Compatibility with common autopilots like PX4 and Ardupilot, useful for Software-In-The-Loop (SITL) and Hardware-In-The-Loop (HITL) testing
Multiple Vehicles	Capability to concurrently simulate vehicles
Heterogeneity	Integration posibilities with other platforms
Sensors	Integration support for common sensors (e.g., cameras, IMUs, GPS, LIDAR, optical flow)
UAV Models	Support of common UAV models and ease of integrating new models
Simulation Speed	Real-time speed and ability to run in super real-time, crucial for learning applications
APIs	Compatibility with programming languages, middleware like ROS, and package such as OpenAI Gym (now Gymnasium)
Integration	Ease of getting started and development, license type, and maintenance status of the software

Table 2.1: Selection Criteria for UAV Simulators [18]

It can be seen that there is a remarkable improvement in this field: a considerable amount of simulators were created over the course of the last years, while some software (e.g., Microsoft AirSim) is now already outdated. In [16] and [17] Microsoft AirSim is fully supported, on the contrary in [18] it is outdated.

[18] also compares some simulators on supporting a variety of vehicle types. This thesis is focused on UAVs primarily.

In [18], several simulators specialized in swarm systems are explored: gym-pybullet-drones, QuadSwarm, Potato, CrazyChoir, Crazyswarm2, CrazySim, sim_cf2.

Another broad category is software focused on Machine Learning or Reinforcement Learning. Such software should be able to run a large amount of instances at once to enable training of models. On top of that, it is often expected to have photorealistic visual rendering because computer vision algorithms account for a large proportion of ML algorithms.

As we can see from above, each paper mentions Gazebo, an open-source, generic simulator, which is one of the most popular.

Since the beginning of the full-scale russian war against Ukraine, UAVs have become increasingly popular for military tasks, among which there are identification, surveillance, reconnaissance as well as payload transportation. This led to development of different tools for UAVs which should be tested inside a simulator first. That's why there is a demand for research of simulators with a bias towards military use of UAVs. There are already workings on new software with the specified intention, for example, Ukrainian Fight Drone Simulator [20], AutoTrans [10].

Simulator	Paper [17] **2021**	Paper [16] **2018**	Paper [18] **2024**	Paper [19] **2018**	Paper [21] (Swarms) **2023**
Gazebo	+	+	+	+	+
AirSim	+	+	+	+	-
Flightmare	+	-	+	-	-
Webots	+	-	+	-	+
jMAVSim	+	+	-	+	-
X-Plane	-	+	-	-	-
FlightGear	-	+	-	-	-
UE4Sim	-	+	-	-	-
Isaac	-	-	+	-	-
CoppeliaSim	-	-	+	-	+
FlightGoggles	-	-	+	-	-
gym-pybullet-drones	-	-	+	-	-
RotorTM	-	-	+	-	-
MATLAB UAV Toolbox	-	-	+	-	+
Morse	-	-	-	+	+
New Paparazzi Simulator	-	-	-	+	-
HackflightSim	-	-	-	+	-
RflySim	-	-	-	-	+
ARGoS	-	-	-	-	+
OMNET++	-	-	-	-	+
AVENS	-	-	-	-	+

Table 2.2: Comparison of covered UAV simulators in five research papers

Table 2.2 demonstrates which UAV simulations systems are reviewed in each of the five research papers from different years. All of them cover simulators in general, with a notable exception of the paper [21] which concentrates on UAV swarms. Only Gazebo was mentioned in all five papers making it the most popular simulators. There are also simulation systems discussed in three or four of them. The majority, however, is presented only in one paper.

This list of papers might be a good starting point to familiarize oneself with the topic at the beginner level.

Chapter 3

UAV Simulators Overview

3.1 Gazebo

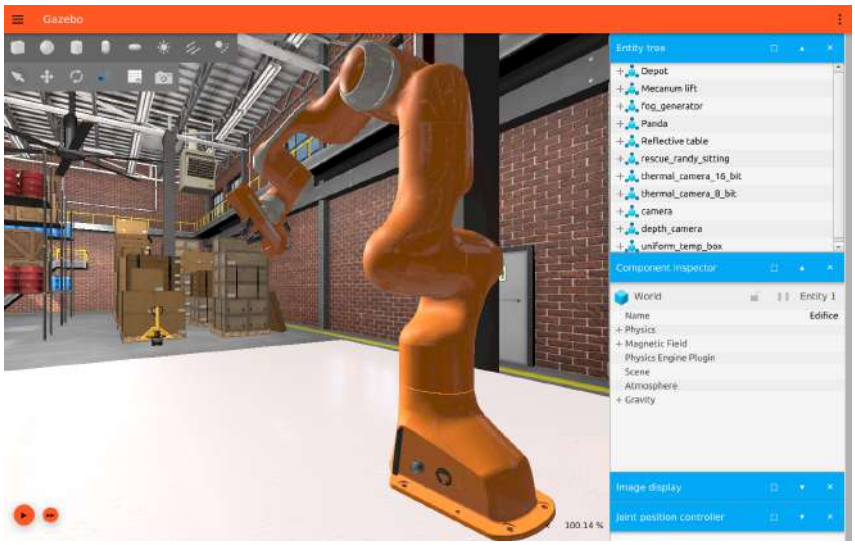

Figure 3.1: Gazebo Simulator [22]

Gazebo is by far the most popular simulator. It is mentioned in all the reviewed papers related to UAV simulators' comparison. Gazebo is a collection of open source software libraries designed to simplify development of high-performance applications. [23] Gazebo [24] is a simulator implemented at the University of Southern California since 2002; since 2009 it has been used with ROS [25]. [16] In 2017, development forked into two versions, known as "Gazebo", the original monolithic architecture, and "Ignition", which had moved to become a modernized collection of loosely coupled libraries. Following a trademark obstacle in 2022 regarding their use of the name "Ignition", Open Robotics took the opportunity to switch the version names, dubbing the original fork "Gazebo Classic" and the new, modern fork "Gazebo". [23] The older version (Gazebo Classic) reaches its end of life in 2025. [23] At the same time, different versions continue to be supported and further developed.

Release List

Name	Date	EOL date	Notes
Jetty	Sep, 2025	Sep, 2030	LTS
Ionic	Sep, 2024	Sep, 2026	
Harmonic	Sep, 2023	Sep, 2028	LTS
Garden	Sep, 2022	Nov, 2024	EOL
Fortress	Sep, 2021	Sep, 2026	LTS
Edifice	Mar, 2021	Mar, 2022	EOL
Dome	Sep, 2020	Dec, 2021	EOL
Citadel	Dec, 2019	Dec, 2024	EOL
Blueprint	May, 2019	Dec, 2020	EOL
Acropolis	Feb, 2019	Sep, 2019	EOL

Figure 3.2: Gazebo Release List [26]

The detailed comparison of features and differences between Gazebo and Gazebo Classic can be found in [27]. This simulator supports PX4 SITL and HITL, as well as Ardupilot SITL and HITL.

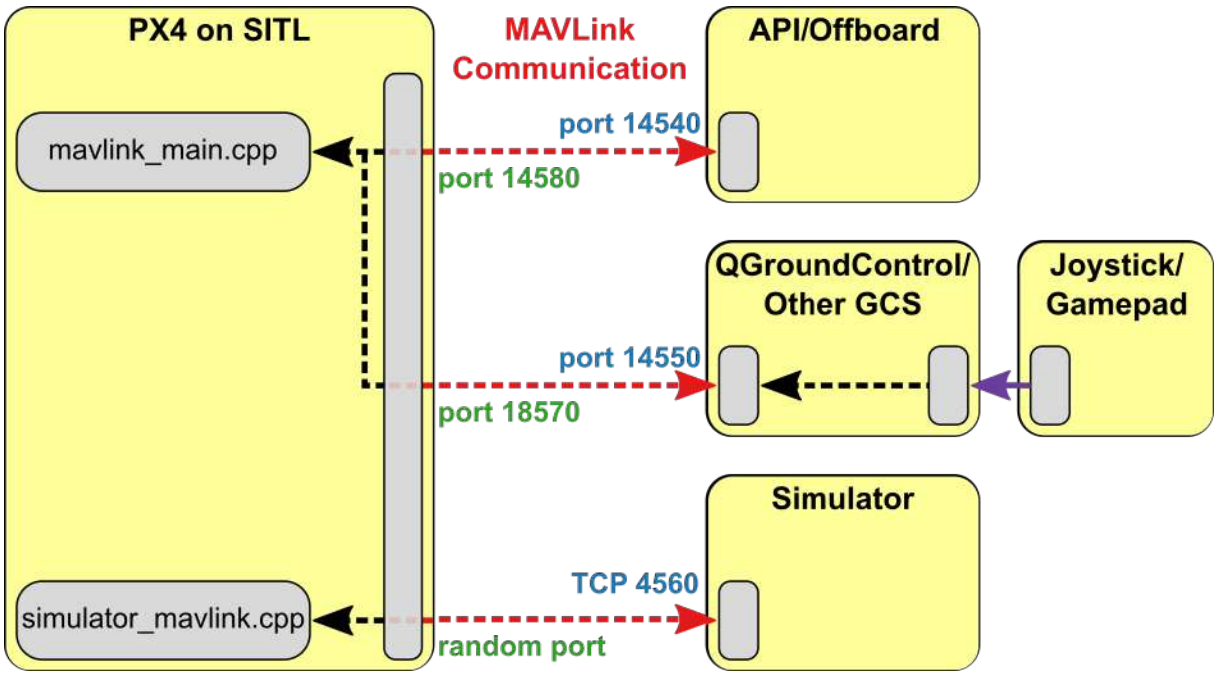

Figure 3.3: PX4 on SITL [28]

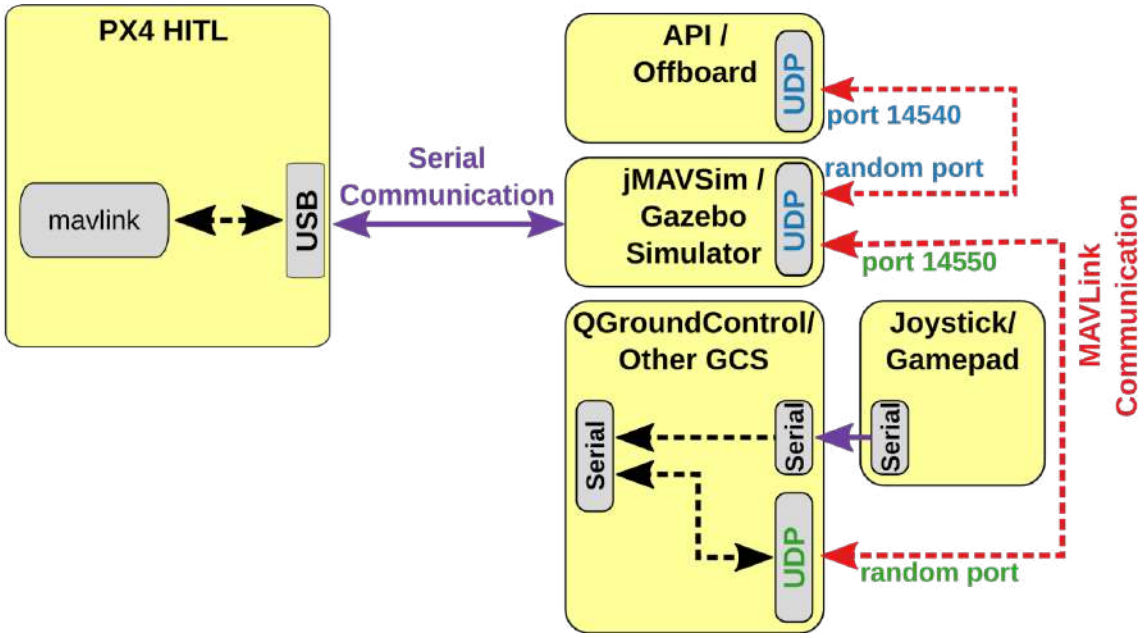

Figure 3.4: PX4 on HITL [13]

One of the disadvantages of Gazebo is its lack of realistic rendering. It might be necessary for tasks that involve computer vision, object detection or object tracking, path planning and obstacle avoidance, collision-free navigation, drone racing and agile flight etc.

Gazebo can be launched on Linux (best used with Ubuntu) and MacOS. [18]

According to [29] and [30], to launch a Gazebo simulator, you will need a computer that has:

- A dedicated GPU with 1 GB or more graphics memory (Nvidia cards tend to work well in Ubuntu)
- A CPU that is at least an Intel I5, or equivalent,
- At least 20 GB free disk space
- 4 GB or more of RAM

Throughout the development of this thesis, Gazebo Harmonic was installed on 2 machines. One of them had the MacOS Sonoma 14.5 operating system. Another one was running Windows 11.

Hardware characteristics of the first machine (MacOS):

- Operational system: MacOS Sonoma 14.5
- RAM: 16 GB
- Total drive memory: 1 TB

- Free drive memory: 386 GB
- CPU: Apple M2 Pro
- GPU: Apple M2 Pro
- Total number of cores: 10 (6 productive and 4 efficient)

Hardware characteristics of the second machine (Windows):

- Operational system: Windows 11
- RAM: 8 GB
- Total drive memory: 512 GB
- Free drive memory: 102 GB
- CPU: Intel Core i5 1135G7
- GPU: Intel Iris Xe Graphics
- Total number of cores: 4

The second one had no dedicated GPU. In addition, Gazebo was running through Windows Subsystem for Linux (WSL) with Ubuntu. This resulted in a substantial slowdown in Gazebo's performance, reducing it to 2 frames per second. At the same time, the instance running on MacOS was operating at a significantly faster speed of 60 frames per second.

3.2 Flightmare

Flightmare is a modular and flexible quadrotor simulator that is mainly composed of two separate components: a photo-realistic rendering engine built with the Unity Editor and a quadrotor dynamics simulation. [31]

Figure 3.5: Flightmare Simulator [31]

This simulator is open-source as well. The last commit in the repository [32] was published on May 15, 2023. Whereas the first commit was made on Jul 14, 2020. So the original repository is not actively supported at the present time. Also there are 365 forks of this repository. However, none of them has achieved widespread adoption: the most popular one has 4 stars. The original Flightmare repository has 1.1 thousand stars. [32]

The rendering engine can generate realistic visual information and simulate sensor noise, environmental dynamics, and lens distortions with minimal computational overhead. [18]

Flightmare supports IMU, RGB, a depth map, and a semantic segmentation or an instance segmentation [31]. Flightmare is extensively used for ML applications, such as for autonomous drone racing [18]. Flightmare can simulate up to several hundreds of agents in parallel, which is not only useful for multi-drone applications but also enables extremely fast data collection and training, which is crucial, e.g., for developing deep reinforcement learning (RL) applications [31]. Flightmare can be launched only on Linux.

Another potential disadvantage of Flightmare is an absence of HITL and SITL support.

Furthermore, Flightmare lacks support for some sensors, specifically GPS, barometer, sonar, radar etc. [17]

3.3 Microsoft AirSim

AirSim is a simulator published by Microsoft in 2017. It supports photorealistic rendering and is based on the Unreal Engine. One of the main focuses during the creation of AirSim was deep learning/reinforcement learning. Also, the great visuals can be a downside

because AirSim is resource-intensive and hence requires large computing power to run when compared with other simulators [17].

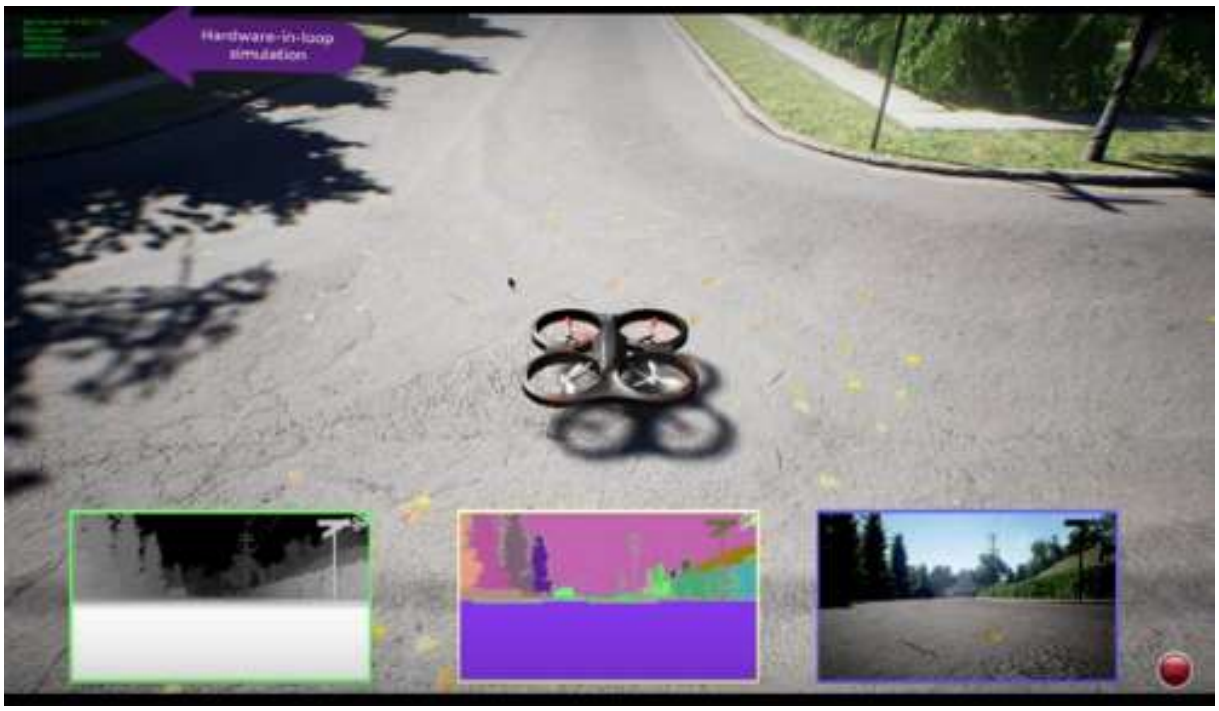

Figure 3.6: Microsoft AirSim Simulator [33]

AirSim simulator provides support to a large range of sensors, including IMU, barometer, GPS etc [34].

In addition, the possibility of interfacing with Mavlink is present meaning that HITL and SITL are possible with AirSim and ArduPilot or PX4 controllers.

The original AirSim is open-source, but will no longer be supported by Microsoft. Their focus has shifted to Project AirSim, which will be released under a commercial license [18].

However, in general, Microsoft AirSim was a quite powerful and popular simulator since it is mentioned in all the papers overviewing UAV algorithm development environments.

3.4 NVIDIA Isaac Sim

Isaac Sim is the newest robotics simulator among those considered. It was first released in 2023. It is developed by NVIDIA and is not an open-source project. Isaac Sim is fully extensible. Isaac Sim facilitates three essential workflows: generating synthetic data for training and fine-tuning robot foundation models, conducting software-in-the-loop testing for robot stacks, and enabling robot learning through Isaac™ Lab. Isaac Sim supports various sensors, including vision-based, RADAR, LIDAR, contact, and IMUs, as well as

custom sensors [35].

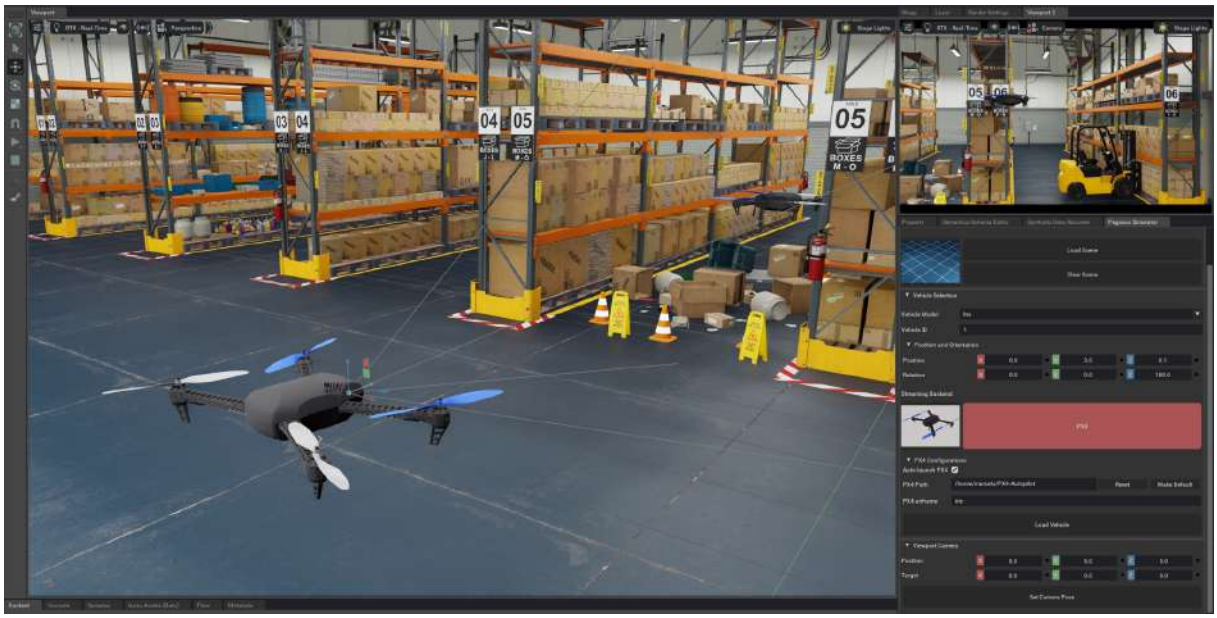

Figure 3.7: NVIDIA Isaac Sim Simulator [36]

NVIDIA Isaac Sim is available for Linux and Windows, whereas MacOS is not supported. Due to its architecture, new libraries and extensions might be constructed on top of Isaac Sim (e.g., Pegasus Simulator [36]). Pegasus Simulator is an extension framework build upon Isaac Sim. It addresses the lack of UAV models available in Isaac Sim, as well as the absence of basic sensors that are typically present on drones. In addition, intergration with the PX4 filmware and the MAVLink communication protocol was implemented. The frameworks aims to be both photorealistic and easy to use within the aerial robotics community.

Also, Paper [18] mentions Isaac Gym as a library for GPU-accelerated Reinforcement Learning (RL) simulations but at the present day it is already deprecated. Using Isaac Lab [37] is recommended instead. Isaac Lab is an open-source lightweight and performance optimized application for robot learning built on the Isaac Sim platform [37].

3.5 Overall Evaluation

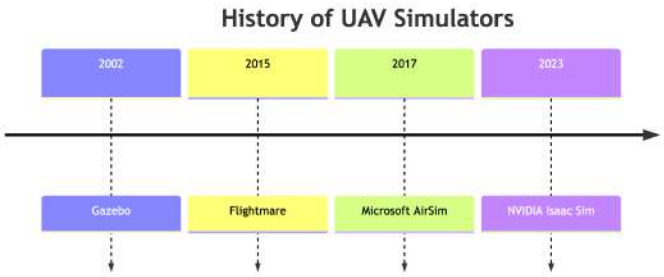

Figure 3.8: Timeline of UAV simulators appearance

Name	First release	Latest update	Linux	Mac	Windows	Open-source	SITL	HITL
Gazebo	2002	09.2025	+	+	WSL	yes	PX4, ArduPilot	PX4, ArduPilot
Flightmare	2020	05.2023	+	-	-	yes	SITL absent	HITL absent
AirSim	2017	07.2022	+	+	+	yes	PX4, ArduPilot	PX4, ArduPilot
Isaac Sim	2023	09.2024	+	-	+	no	SITL absent	HITL absent

Table 3.1: Aerial robotics simulators comparison: Gazebo, Flightmare, AirSim, Isaac Sim

Chapter 4

Analysis of the Distribution of Development Environments and Flight Software

4.1 Approach

To estimate the current state of usage of UAV simulators and flight software, it was determined to use the next scientific search engines: Google Scholar and Semantic Scholar.

Google Scholar is a search engine specializing in academic literature. It includes content from all leading publishers with the exception of Elsevier and the American Chemical Society—as well as from hosting platforms like Highwire and Ingenta. Users can find articles from journals, reports, theses, books, preprints, and more on Google Scholar. When a search is performed, Google Scholar retrieves documents or pages matching the keywords and ranks the results using a proprietary relevance algorithm [38].

Semantic Scholar is an artificial intelligence-enabled search engine, found by the nonprofit Allen Institute for Artificial Intelligence (AI2). It began as a search engine for computer science, geoscience, and neuroscience in 2015. Its purpose is to overcome information overload with the means of automatic learning from text. [39] They index over 200 million academic papers sourced from publisher partnerships, data providers, and web crawls. [40]

By having a look at the amount of the search results for queries, conclusions about the popularity and prevalence of simulators can be derived.

4.2 Rationale for Examining the Pairing of Simulators and Flight Software

Flight control software plays a great role in UAV algorithms development and setting up the environment for testing.

Basically, a UAV simulator generates sensor values (IMU values, as well as camera) and sends them to flight software that interprets them and transforms them into commands for motors.

What is returned from flight software to simulators? Motor commands? Because of that not only simulators popularity was measured. Additionally, their pairing with chosen flight software was also measured in Google Scholar and Semantic Scholar.

Gazebo, Flightmare, Microsoft AirSim, and NVIDIA Isaac Sim were selected as UAV environments.

PX4, Ardupilot, BetaFlight and INAV were selected as flight control software. All of them are open-source and support a wide range of sensors, protocols, and vehicle types.

- **PX4** is a full-featured open-source project, started in 2009. It can run multiple types of vehicles including quadcopters and fixed-wing planes. It is one of the major research platforms for UAVs.

- **ArduPilot** is another widespread autopilot software. It is divided into several branches: Ardupilot for fixed-wing aircraft, Arducopter for multirotors and helicopters, and Ardurover for ground-based vehicles. ArduPilot shares many common features with PX4 (e.g., the use of MAVLink protocol and compatibility with the same software, such as QGroundControl). So both software might be suitable in the same UAV software development projects.

- **BetaFlight** is one of the most popular flight control software for FPVs (First Person View). It aims to maximize drone's agility, speed, and easy tuning. The mentioned features are usefull during drone racing.

- **INAV** is a fork of BetaFlight. It was focused on enhancing autonomous features extending it out of drone racing field. It allows a pilot to create missions like moving to a certain waypoint or returning back to the launch position.

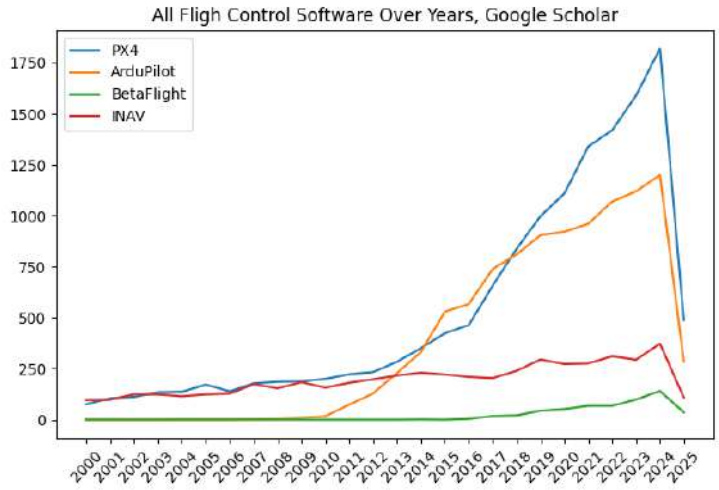

Figure 4.1: Flight Control Software Comparison Over Years, Google Scholar

4.3 Solution

The repository with the source code can be found under the following link: https://github.com/koejdga/flight-software-statistics.

This repository is devoted to the retrieving of mentions' statistics for two entities, namely, UAV simulators (e.g., Gazebo) and flight control software (e.g., ArduPilot).

Mention's amount is retrieved from two academic search engines: Google Scholar and Semantic Scholar.

Google Scholar has no official API, to retrieve query results from it a tool called SerpApi was used. SerpApi is a paid API with a free plan that supports a range of information including the amount of responses for the query, which is desirable in this research. Also, SerpApi handles the legal part of scraping.

In turn, Semantic Scholar has an official API that was utilized in the research.

The search queries conducted to those engine include either only the name of the product (e.g., Gazebo) or the name and the word "uav" appended at the end of the query.

Finally, the statistics come in two variants with the first being the total papers' amount and the second being the papers' amount over the years. Also, to find out usage of specific simulator with specific flight software, queries with such structure were used: UAV simulation flight software name simulator name (e.g., "UAV simulation PX4 gazebo" without parentheses)

4.3.1 The objective of the project

There are two main reasons why this repository might be useful. The first one is the already gathered statistics and visualizations of it The second one are the scripts that enable the gathering of the statistics later again.

The gathered statistics might give someone insights about the popularity of the products overall or over the years.

The comparison between Google Scholar and Semantic Scholar can provide insights about where it is better to search for the relevant information depending on what simulator or flight software is a subject of your interest. Those statistics can give a rough overview.

The search algorithms of both search engines have their peculiarities. One of them might be rounding the number of results. That means that no exact results are represented in the provided plots. Nevertheless, the graphs show tendencies that are present in the field.

4.3.2 Used libraries

- **matplotlib** - to generate plots with graphs for visualizations
- **seaborn** - to generate heatmap plots
- **requests** - to make HTTP requests
- **dotenv** - to load variables from .env file
- **argparse** - parses command-line arguments

4.4 Analysis of the total amount of data

The analysis of total amount of returned responses was conducted with the use of Google Scholar. It indexes a vast range of academic literature and stays up-to-date providing access to recent publications and trends.

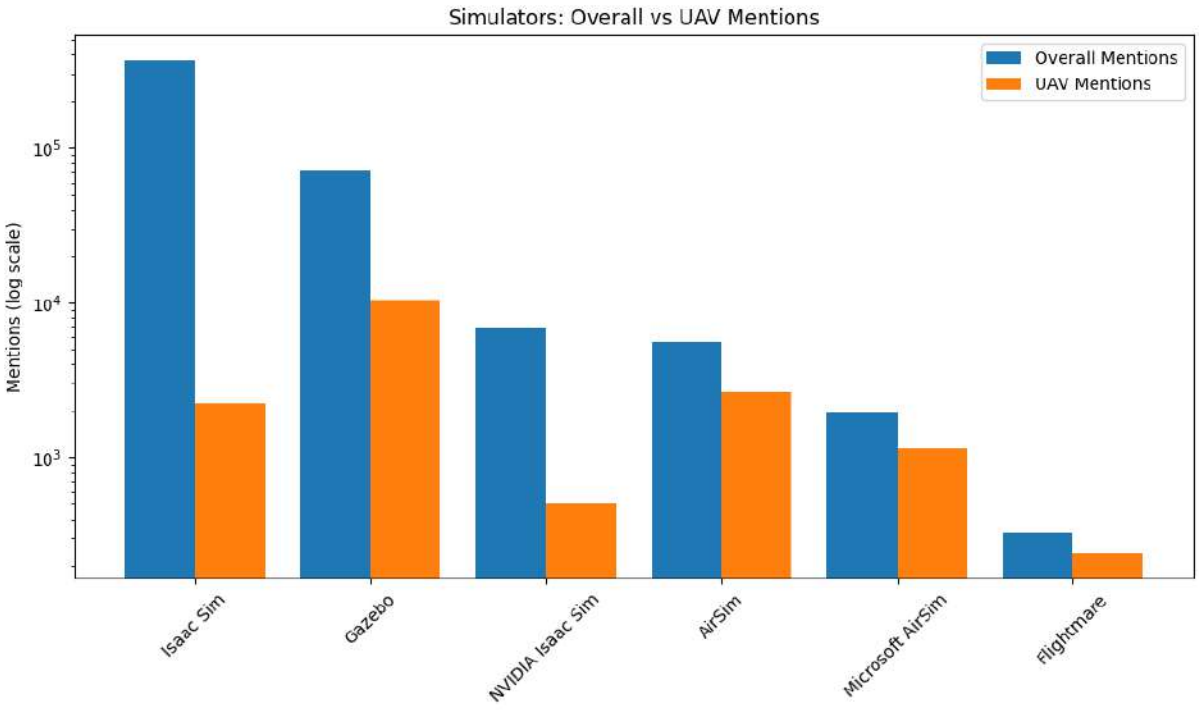

Figure 4.2: UAV Simulators: Overall vs UAV mentions

In Figure 4.2, "Isaac Sim" search showed the biggest amount of returned responses among all the simulators. However, considering that the results for "Isaac Sim" with "UAV" word are about three times smaller, presumable "Isaac Sim" search captured a significant number of irrelevant papers from other scientific fields. The responses for "NVIDIA Isaac Sim" with and without "UAV" word are considerably smaller. Hence they might shed more light and clarity to the actual popularity of Isaac Sim simulator.

At the same time, the difference between searches of "AirSim" and "Microsoft AirSim" differ as well but not to the same extent as with Isaac Sim.

If we neglect "Isaac Sim" search results' amount, Gazebo is the most widespread simulator. Moreover, the amount of results for the query with "UAV" are smaller by less than a half. Supposedly this means that Gazebo simulator is used primarily for the aerial robotics field, UAVs.

Finally, Flightmare takes the last place in this ranking while having several times lower result numbers than other simulators.

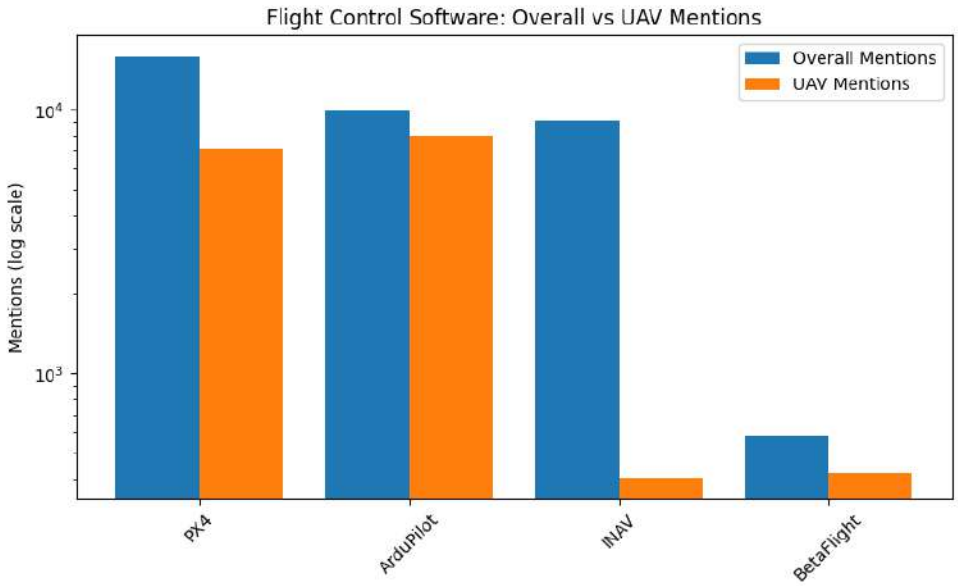

Figure 4.3: Flight Control Software: Overall vs UAV mentions

Figure 4.3 illustrates that PX4 and ArduPilot are the most popular flight control software among the four selected. "INAV" has approximately the same amount of results returned as ArduPilot. However, this appears to be inaccurate. Firstly, similar to the case with NVIDIA Isaac Sim, there is a drastic difference between the amount of results for "INAV" with and without "UAV" word in a query. Secondly, looking at other metrics of popularity, INAV is not that popular as PX4 and ArduPilot. All three software are open-source. PX4 has 9.5 thousand stars on its GitHub repository [41], ArduPilot [42] has 12.1 thousand stars. On the other hand, INAV [43] has only 3.5 thousand. This difference points to the conclusion that irrelevant papers were counted too. In addition, the name "INAV" is an abbreviation. For instance, it can also mean Intraday Net Asset Value in a context of financial markets and trading. Ultimately, BetaFlight has the least amount of mentions in papers in Google Scholar, even though it is rapidly growing and has 9.4 thousand stars on the GitHub repository [44].

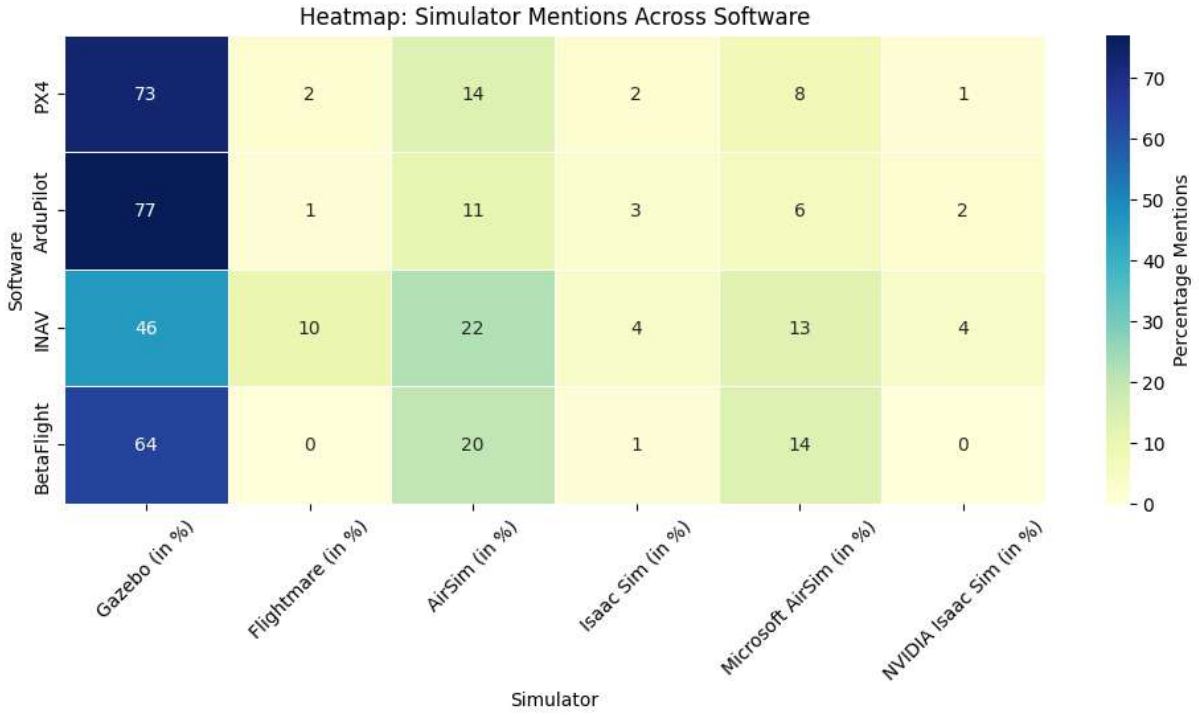

Figure 4.4: Heatmap: Simulator Mentions Across Software

The next step is to look at pairings of robotics simulators and flight control software. Figure 4.4 demonstrates the proportions of papers with the specific flight control software and simulators. From this Figure, it can be seen that Gazebo and PX4 play a significant role for all simulators with PX4 being a bit more widely used. An exception arises in the case of NVIDIA Isaac Sim: it is slightly more commonly coupled with ArduPilot.

Whereas INAV software is met together with Gazebo and Microsoft AirSim, there are no papers about INAV found with Flightmare and NVIDIA Isaac Sim. BetaFlight is, in turn, met with all four simulators.

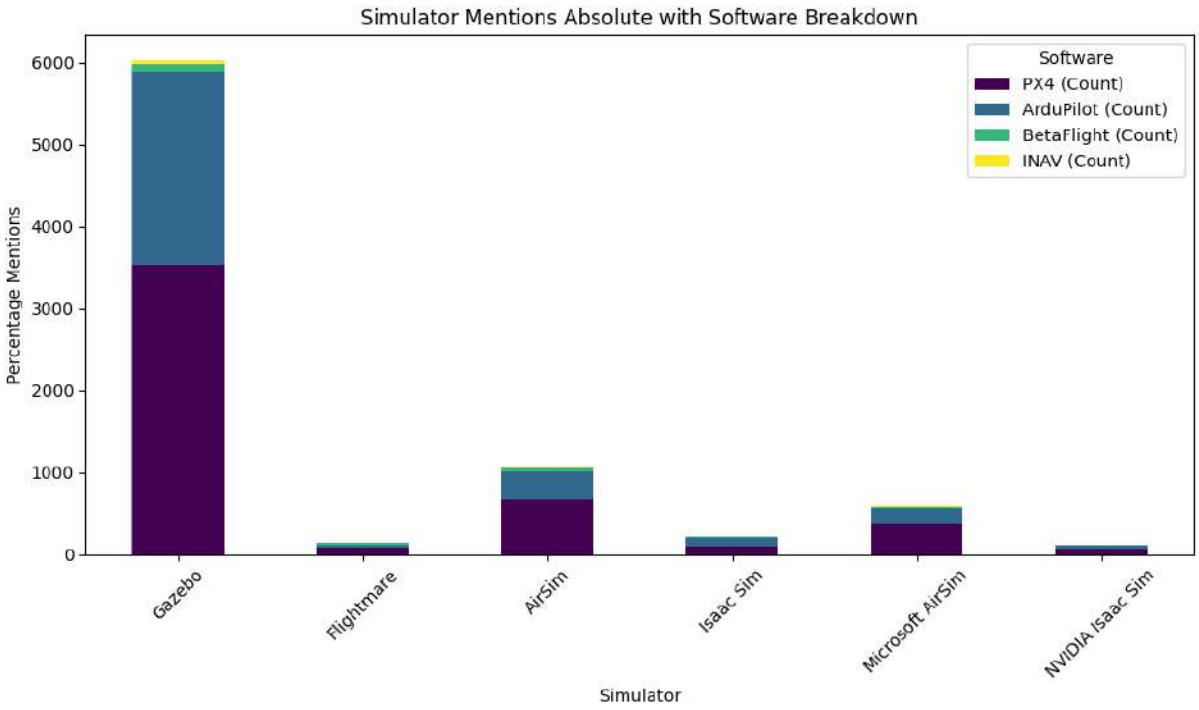

Figure 4.5: Absolute Simulator Mentions with Software Breakdown

The previous plot showed a relative proportions of the results. This means that each simulator has 100% and the shares of flight control software is illustrated. Nevertheless, that does not represent the absolute amount of shares for different flight control software with the simulators.

An examination of Figure 4.5 reveals that actually Gazebo-PX4 and Gazebo-ArduPilot prevail among the considered pairings in papers. Flightmare and NVIDIA Isaac Sim have decently low amounts of papers. Conversely, Microsoft AirSim has higher results. This leads to a conclusion that it was popular one time, even though the project is reaching end of life in the coming year [45].

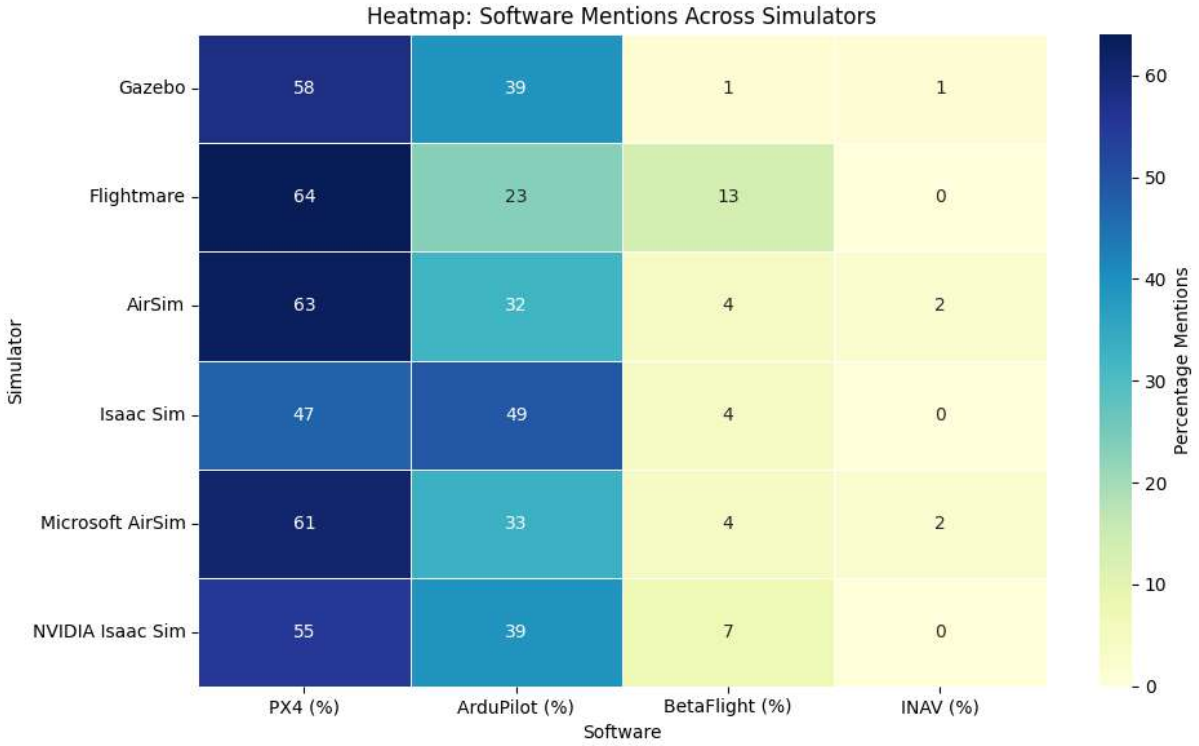

Figure 4.6: Heatmap: Software Mentions Across Simulators

Analogously to Figure 4.4, Figure 4.6 gives a look on how different simulators are spread among flight control software. The same pattern is observed as earlier: Gazebo is the predominant simulators with the flight software, AirSim also takes a decent share (AirSim and Microsoft AirSim represent the same simulator). Meanwhile, Flightmare and NVIDIA Isaac Sim remain unpopular.

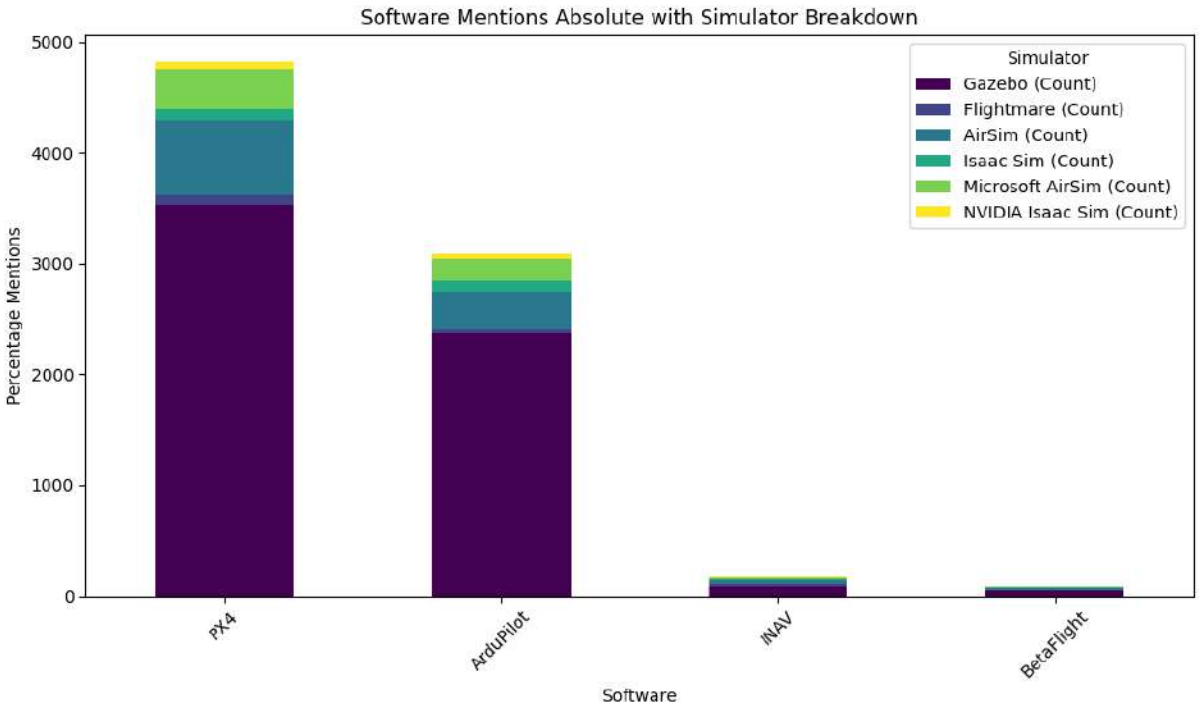

Figure 4.7: Absolute Software Mentions with Simulator Breakdown

In Figure 4.7, the remarkable popularity of PX4 and ArduPilot are also seen, while INAV and BetaFlight are significantly less widespread.

4.5 Analysis of the amount of the papers over the years

In the previous section, the analysis was carried out upon the data from all the years together. However, it often makes sense to examine the data throughout the years. This might reveal additional insights.

In this thesis, the statistics was collected over the period 2000–2025. The year 2000 was chosen as the starting point because examining earlier dates was not meaningful. The oldest simulator considered, Gazebo, was published in 2002, which explains why no activity is expected before 2000.

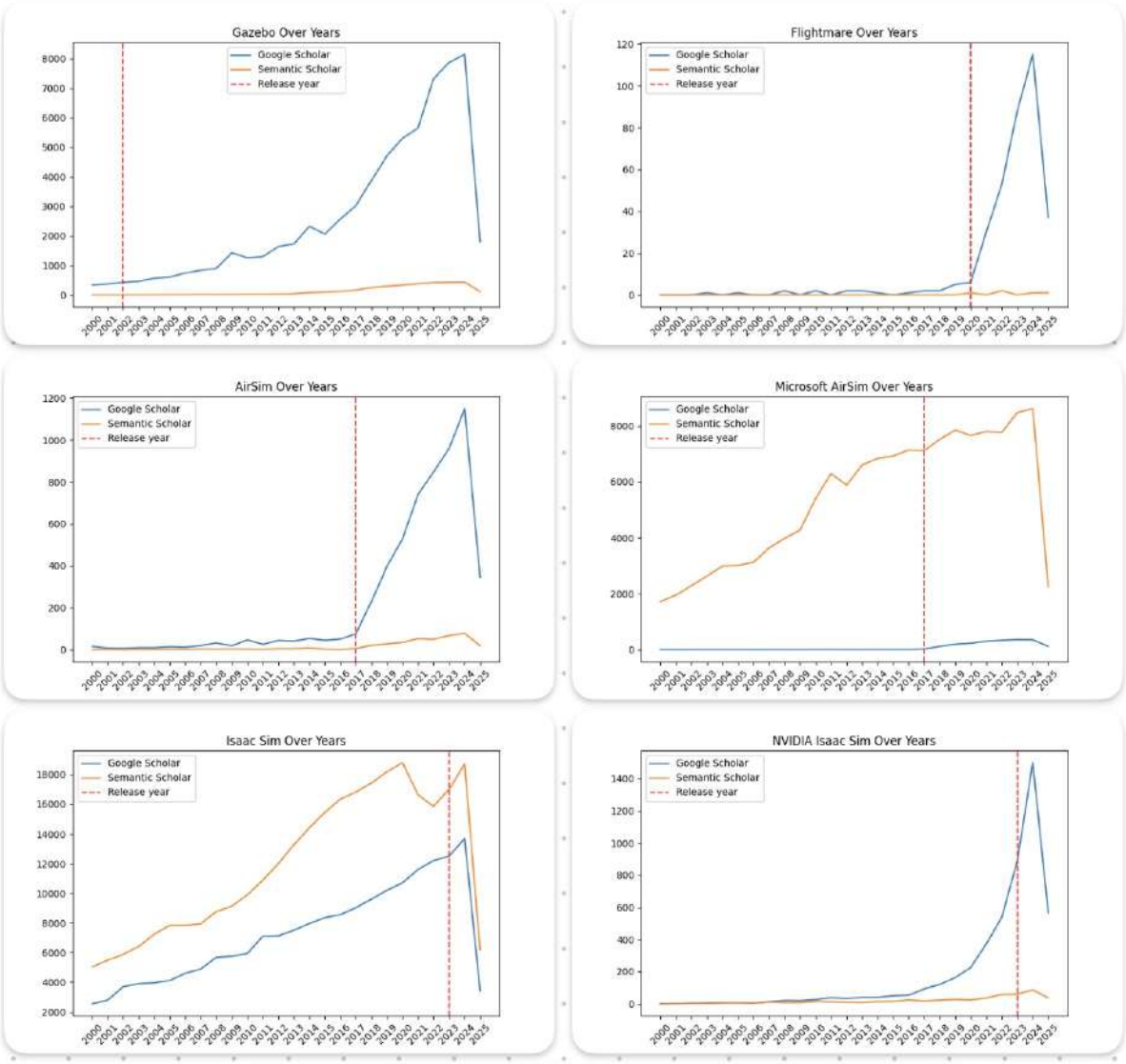

Figure 4.8: Comparison of Simulators over the Years in Google Scholar and Semantic Scholar

Figure 4.8 illustrates how the amount of responses to queries that contain the name of the simulator looked changed throughout the years. The red dashed vertical line points out the release year of the simulator. Partially, we can notice that the amount of some simulators mentions grew substantially. "Gazebo", "Flightmare", and "AirSim" searches fall into this category. Meanwhile, "Microsoft AirSim" and "Isaac Sim" were returning immense result numbers long before the publish year of the corresponding simulators and even in the first year of gathered statistics (2000). Finally, the situation with "NVIDIA Isaac Sim" differs slightly. The growth starts in 2017 despite the fact that it was released in 2023.

This visualization also shows that Google Scholar has more papers about the considered simulators than Semantic Scholar, with exception in "Microsoft AirSim" and "Isaac Sim" queries. This difference may be attributed to Semantic Scholar being newer than

Google Scholar and lacking the same scale of scientific database.

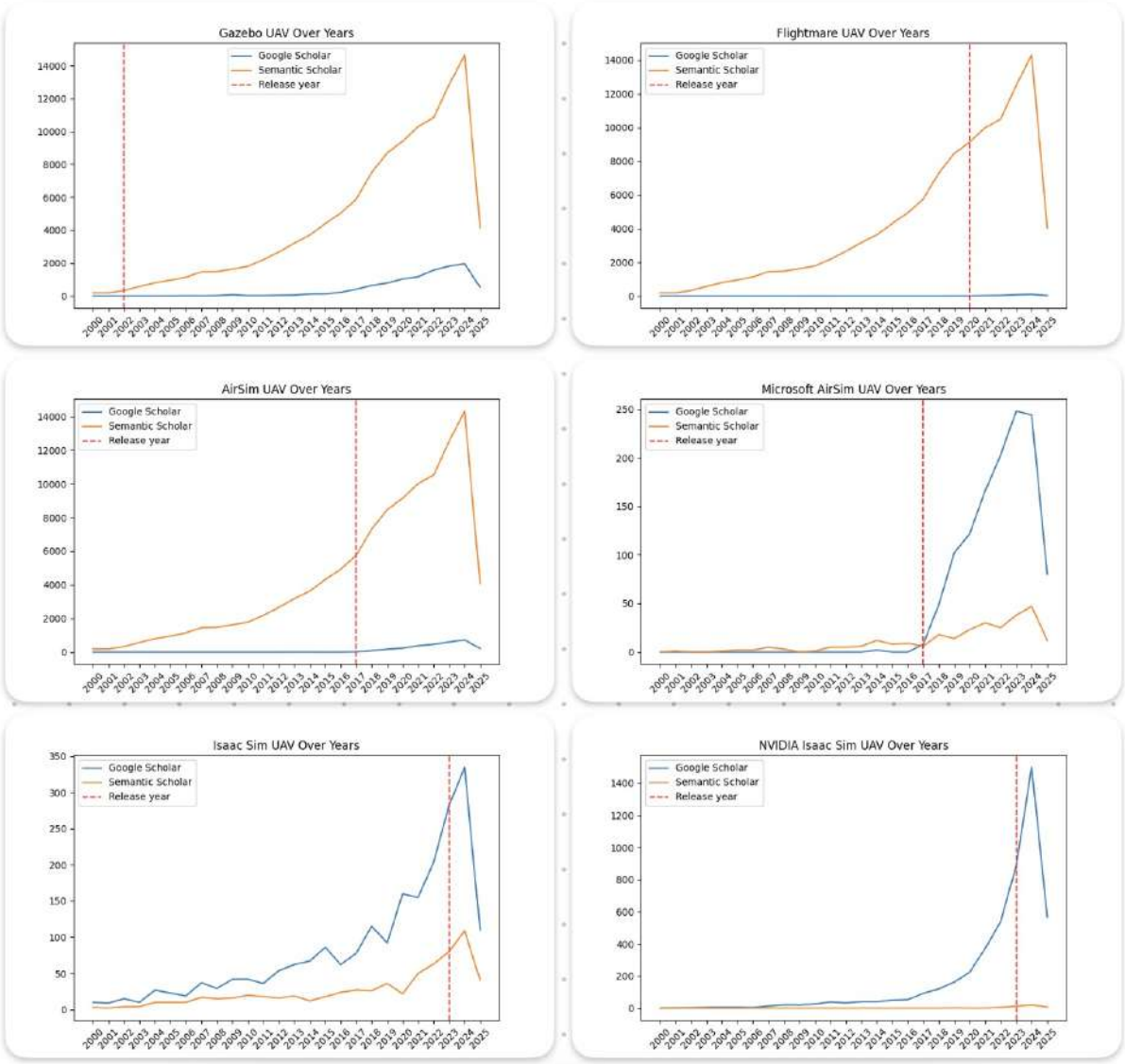

Figure 4.9: Comparison of Simulators with "UAV" in query over the Years in Google Scholar and Semantic Scholar

In Figure 4.9, the similar situation is observed. The amount of papers grows drastically each year. One peculiar observation about Sematic Scholar is that the amount of results for queries "Gazebo UAV", "Flightmare UAV", and "AirSim UAV" are several times bigger than the ones from Google Scholar. The results from Semantic Scholar peak at around 14 thousand for the mentioned queries. Also, the results for these simulators with the word "UAV" appended are several thousands bigger than the results for the same simulators without the word "UAV". This suggests irregularities in the search and indexing algorithm of Semantic Scholar.

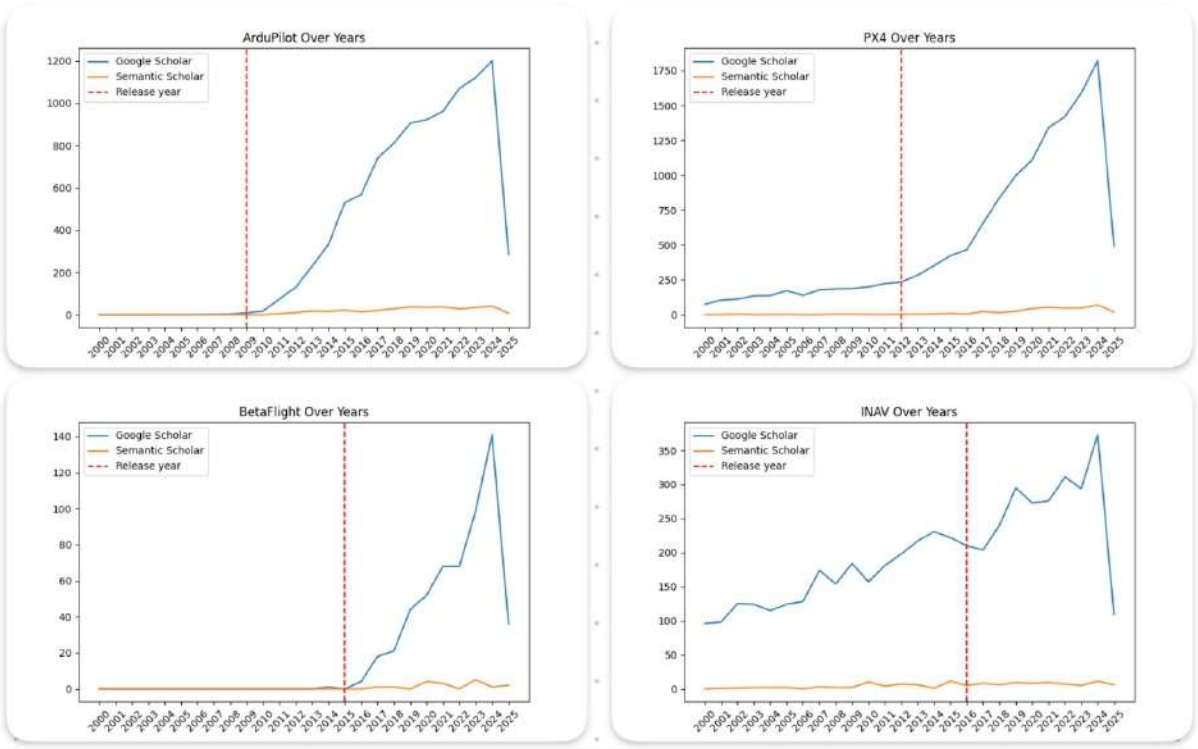

Figure 4.10: Comparison of Flight Control Software over the Years in Google Scholar and Semantic Scholar

Figure 4.10 reveals the similar statistics over the years for the flight control software. The trend of gaining popularity remains here as well. Graphs for ArduPilot and BetaFlight show a marked increase starting after their respective release years. Conversely, this pattern is not observed for PX4 and INAV, which have recorded mentions even before their official release years. This discrepancy may also reflect the presence of irrelevant or noisy data.

Google Scholar has more returned responses in these queries than Semantic Scholar.

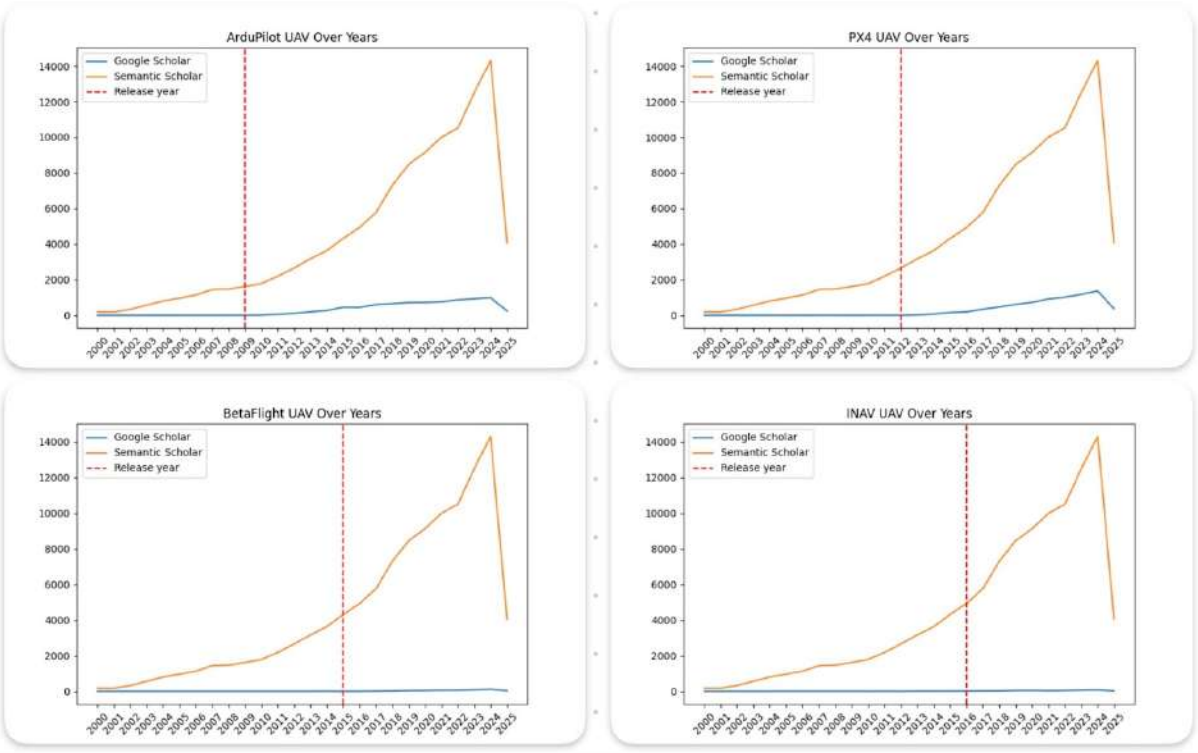

Figure 4.11: Comparison of Flight Control Software with the "UAV" word in a query over the Years in Google Scholar and Semantic Scholar

In Figure 4.11, a pattern can be observed in which the number of Semantic Scholar results for all four flight control software is quite similar, with each showing substantial growth over the years. Additionally, the result counts from Semantic Scholar are several times higher than those from Google Scholar.

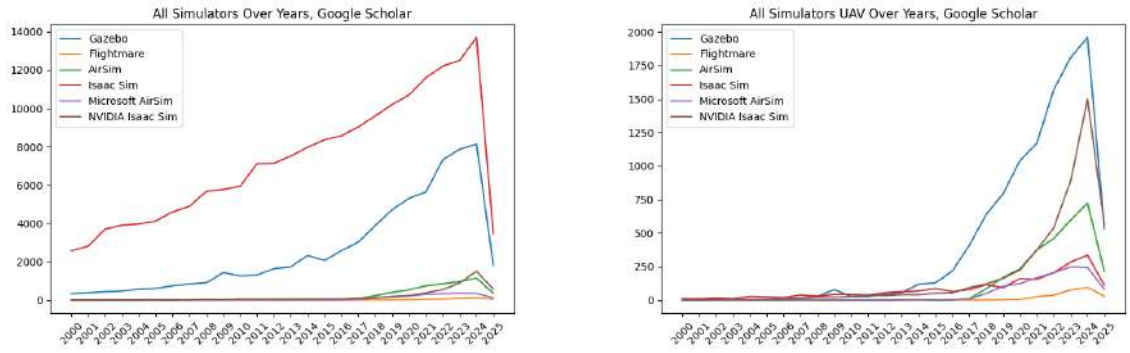

Figure 4.12: Comparison of All Simulators with (right) and without (left) the "UAV" word in a query over the Years (Google Scholar)

Figures 4.12 show more clearly how simulators are comparable with each other.

Chapter 5

Identifying the Research Gap

As it could be seen from the overview above, the most in demand simulator is Gazebo. That is why it made sense to look for research gaps in the Gazebo field first.

One Gazebo environment is called a Gazebo world. It is an XML file with *.world* or *.sdf* extension. The term is used to describe a collection of robots and objects (such as buildings, tables, and lights), and global parameters including the sky, ambient light, and physics properties [46]. Physics behaviour, the weather conditions, positions and rotations of objects etc. are defined inside this file.

The model element defines a complete robot or any other physical object. An SDFormat file can consist of just a model. A world can contain many models [47]. Properties and behaviour of a Gazebo model are declared inside a <model> element in an SDFormat file.

During the research, 18 repositories were found open-source. Mostly, they were obtained from GitHub, at the same time several worlds were located on the separate websites. Some of them are created by gathering the already existing one, so there was some repetition present in models and worlds.

Most of them were published more than 4 years ago. One share of the worlds was developed for Gazebo Classic, which reached its end of life in January 2025. The worlds published by companies, like PX4, ArduPilot, AWS, and Clearpath [62] (a Canadian company that performs robotics research) receive updates more often. The repositories have commits made in 2024 and 2025 years.

In total, 234 world files were accumulated, 164 of them are unique. Simultaneously, analysis of the Gazebo models was also conducted. In total, 1870 model files were accumulated, with 1029 of them being unique. The further analysis was carried out on unique Gazebo worlds and Gazebo models. Both worlds and models files are XML files, where the contents of an entity are declared and then processed by Gazebo during launching. SDF format specification can be found here `http://sdformat.org/spec?ver=1.12`. To estimate the complexities of the found Gazebo worlds and models, the amount of different entities inside were counted.

Number	Latest update	Worlds count	Gazebo version	Source
1	03.09.2020	50	Gazebo new (not classic)	[48]
2	–	7	7	[49]
3	06.12.2020	20	5+	[50]
4	15.09.2023	13	Gazebo new (not classic)	[51]
5	18.04.2019	4	–	[52]
6	29.11.2023	8	–	[53]
7	14.05.2021	12	Gazebo 2, 5, 7, 9, 11	[54]
8	11.02.2021	16	Gazebo 9, 11	[55]
9	25.09.2021	12	Gazebo 11	[56]
10	07.03.2025	10	Gazebo new (not classic)	[57]
11	26.07.2021	51	–	[58], [59]
12	13.05.2024	5	Gazebo 7, 9	[60]
13	11.05.2025	5	Gazebo Garden, Harmonic, Ionic	[61]

Table 5.1: List of resources taken into the Gazebo worlds' dataset

5.1 Considered Entities

- **Model** - defines a complete robot or any other physical object. A model SDF file and a world SDF file can contain zero or many models nested inside them [47].
- **Include** - includes resources from a URI. Included resources in the Gazebo worlds can only contain one 'model', 'light' or 'actor' element. The URI can point to a directory or a file. If the URI is a directory, it must conform to the model database structure. Included resources in the Gazebo models can be used to nest models. The included resource can only contain one 'model' element. One file might have zero or many <include> elements [47].
- **Plugin** - is a dynamically loaded chunk of code. It can exist as a child of the world or model. Zero or many plugins can be present in one world or model [47].
- **Joint** - a joint connects two links with kinematic and dynamic properties. By default, the pose of a joint is expressed in the child link frame [47].
- **Link** - a physical link with inertia, collision, and visual properties. A link must be a child of a model, and any number of links may exist in a model [47].
- **Sensor** - the sensor tag describes the type and properties of a sensor. A link and joint may contain zero or many sensors [47].

5.2 Gazebo Worlds Analysis

The more models in a world there are, the more complex the world is. Firstly, in the research the amount of <model> elements were counted. Then the amount of worlds, in which such an amount of models was found. The same principle was applied to other elements' analysis.

42% of the worlds do not contain <model> elements. They are neglected in Figure 5.1. It can be seen in Figure 5.1 that a decent amount of worlds (12%) contain only one model. Also, 2 and 3 models are met in 7 worlds each. Other than that, in the collected dataset, worlds featuring dozens or even hundreds of models were encountered.

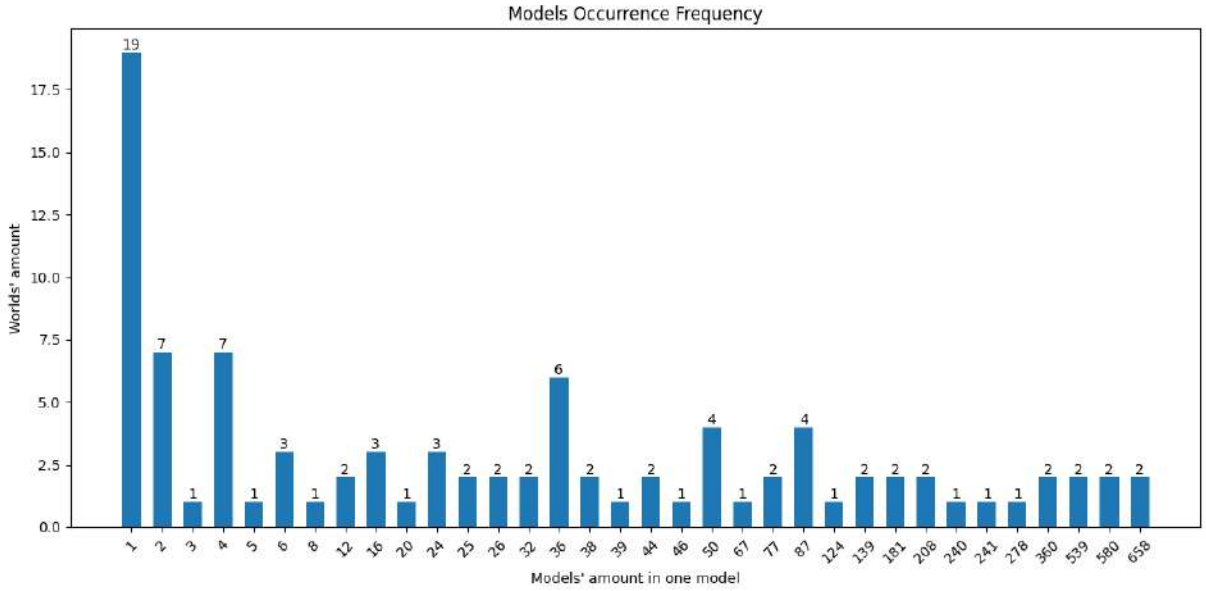

Figure 5.1: Models Occurrence Frequency in Worlds

Since it often makes sense to keep code or markup modular and to separate concerns, Gazebo models are often put in separate files or folders. This facilitates the reuse of code and helps to avoid duplication. To use models from a local file or folder, or from a URL on the Internet, <include> elements are used. That is why their occurrence frequency was calculated. The results are similar to those with the <model> elements but with a slight shift. 36% of all worlds contain no <include> elements. 33% of all worlds contain from 1 to 4 elements. The rest 31% consist of more than 4 models, peaking at 542 <include> elements in one world.

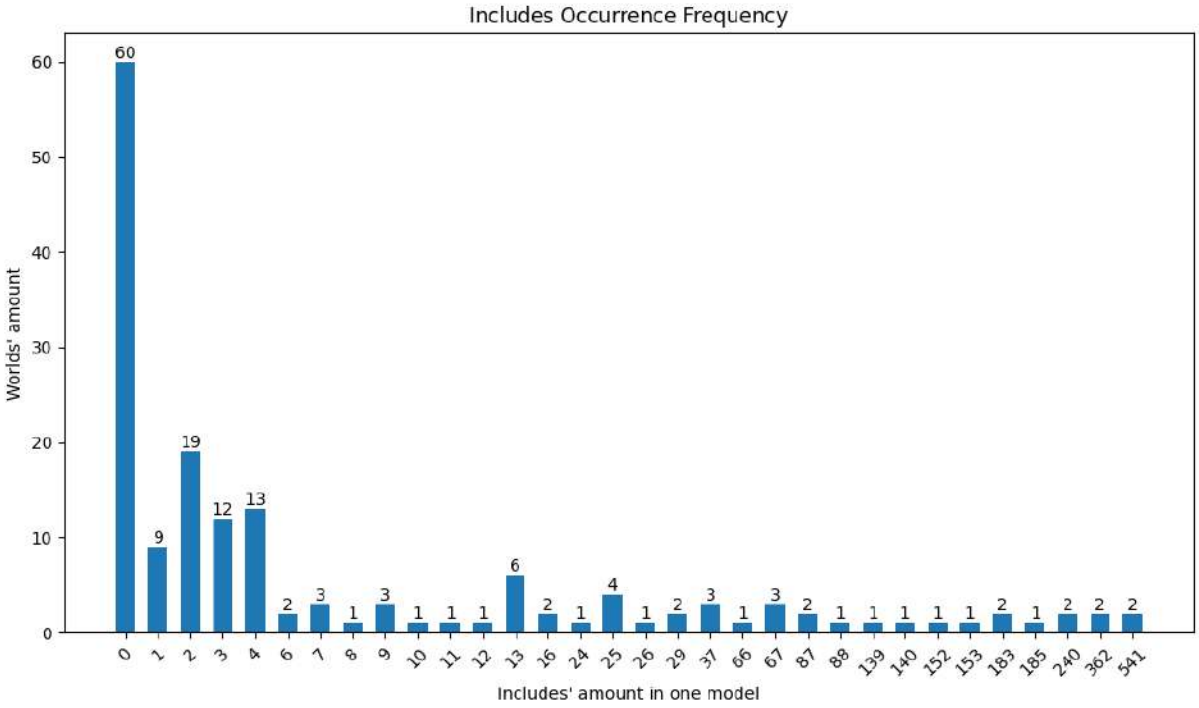

Figure 5.2: Includes Occurrence Frequency in Worlds

Model and include elements are semantically similar because they both represent a model inside the world. Therefore, it makes sense to look at the occurrence frequency of them both together. In Figure 5.3, there are no worlds that contain neither <model>, nor <include> elements. 40% of the worlds contain from 1 to 4 models (with includes). This means that most worlds are not very complex, even though it also depends on included models' complexity, which was not estimated in this step.

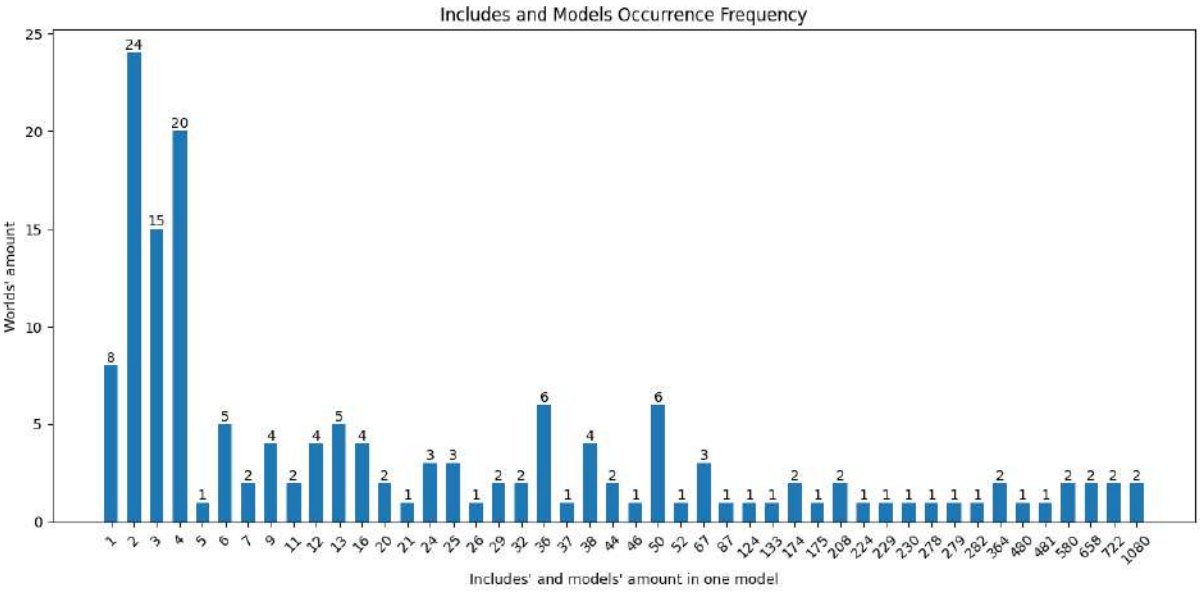

Figure 5.3: Includes and Models Occurrence Frequency in Worlds

Moving further, plugins, joints, and sensors occurrence frequencies were computed. For the results to be more representative, plugins that are present in the models that were added with the <include> elements were also added up. For that, each world was scanned for <include> tags and the corresponding SDF files for included models were parsed in order to find all the plugins, joints, and sensors. If this step was skipped, a decent amount of plugins, joints, and sensors would be neglected in the analysis leading to less accuracy.

As it can be seen in Figure 5.4, the vast majority (74%) of the worlds do not include any plugins. This leads to an assumption that those worlds are rather static and simple, without moving objects etc.

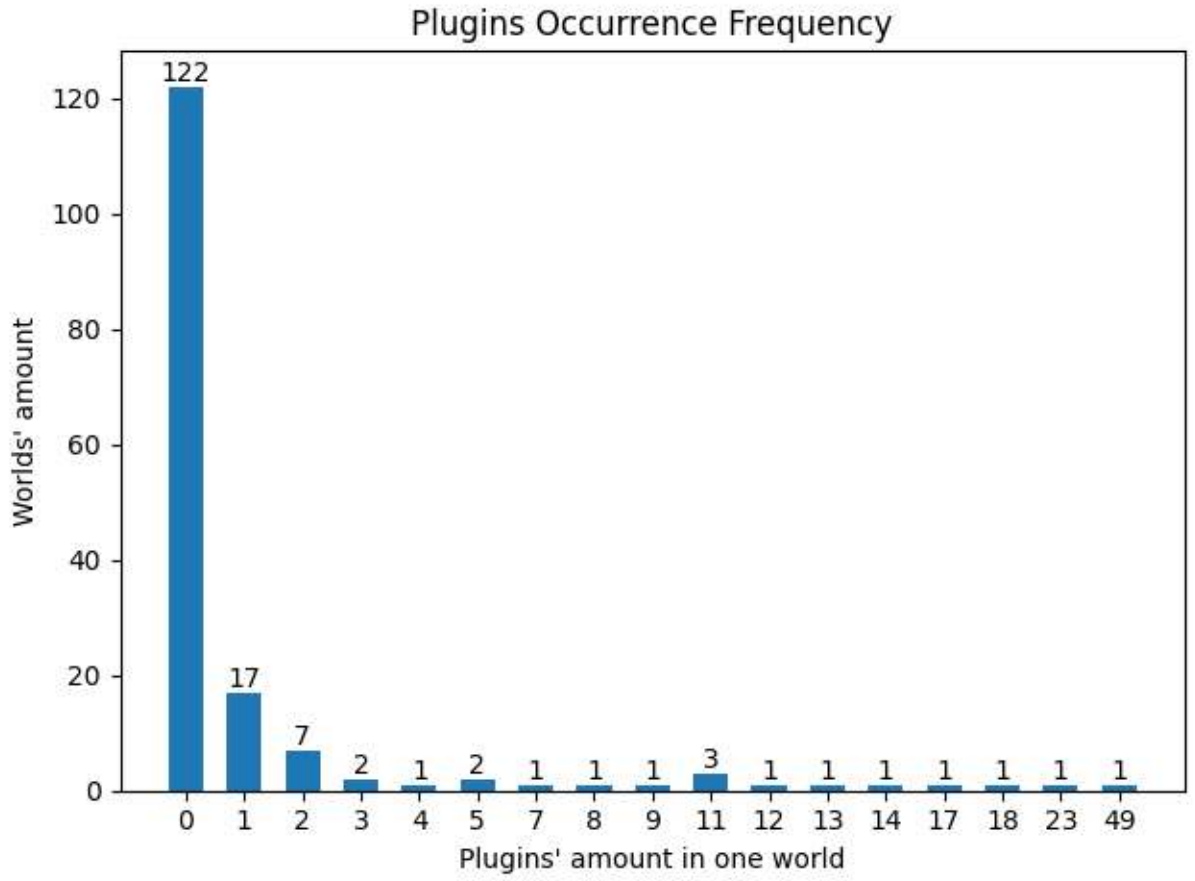

Figure 5.4: Plugins Occurrence Frequency in Worlds

On Figure 5.5, the rating of plugins with their popularities is shown. *gz-sim-lift-drag-system* is by far the most popular plugin. It occurs in 30 different Gazebo worlds. Other top plugins are *librotors_gazebo_ros_interface_plugin.so*, *libRandomVelocityPlugin.so*, and *gz-sim-apply-joint-force-system*, apperaring in 20, 14, 14 worlds, accordingly.

The LiftDrag system (*gz-sim-lift-drag-system*) computes lift and drag forces enabling simulation of aerodynamic robots. Lift is the force that pushes an object upwards against gravity. Drag is the force that resists or slows down the object's forward motion. These

forces are crucial in aerodynamics because they determine how air moves around and affects a flying robot (e.g., a UAV). By computing these forces, the system allows for realistic simulation of how aerodynamic robots behave in flight, helping to predict their movement and performance under different conditions.

gz-sim-apply-joint-force-system applies a force to the first axis of a specified joint.

The Random Velocity Plugin *libRandomVelocityPlugin.so* allows you to assign a random velocity to a link in the world.

Most plugins are used to enable robot movement by simulating forces acting on them, similar to real-life conditions. Plugins such as *gz-sim-imu-system* are specifically designed for simulating aerial robots, for example, UAVs.

All the plugins may be divided in 3 categories:

- related to RotorS [63], a modular Micro Aerial Vehicle (MAV) simulation framework, which is based on Gazebo. Their names start with "librotors" and are written in snake case.
- related to Gazebo Classic. Their names start with "lib" and are written in camel case. Also they can start with "libgazebo" and are written with snake case.
- related to new Gazebo, Their names start with "gz-sim" and are written in kebab case.

In total, 39 distinct plugins were identified in Gazebo worlds from the dataset. 11 of them (**28%**) are related to new Gazebo Simulator. Also, other 11 (**28%**) are related to RotorS. Finally, the rest 44% are associated with Gazebo Classic.

However, when looking not only at their absolute amounts but also at the amounts of worlds each plugin was found in, the picture is different. 91 out of 202 times (45%) plugins for new Gazebo were found. 30% account for plugins correlated with RotorS, and only 25% constitute plugins for Gazebo Classic.

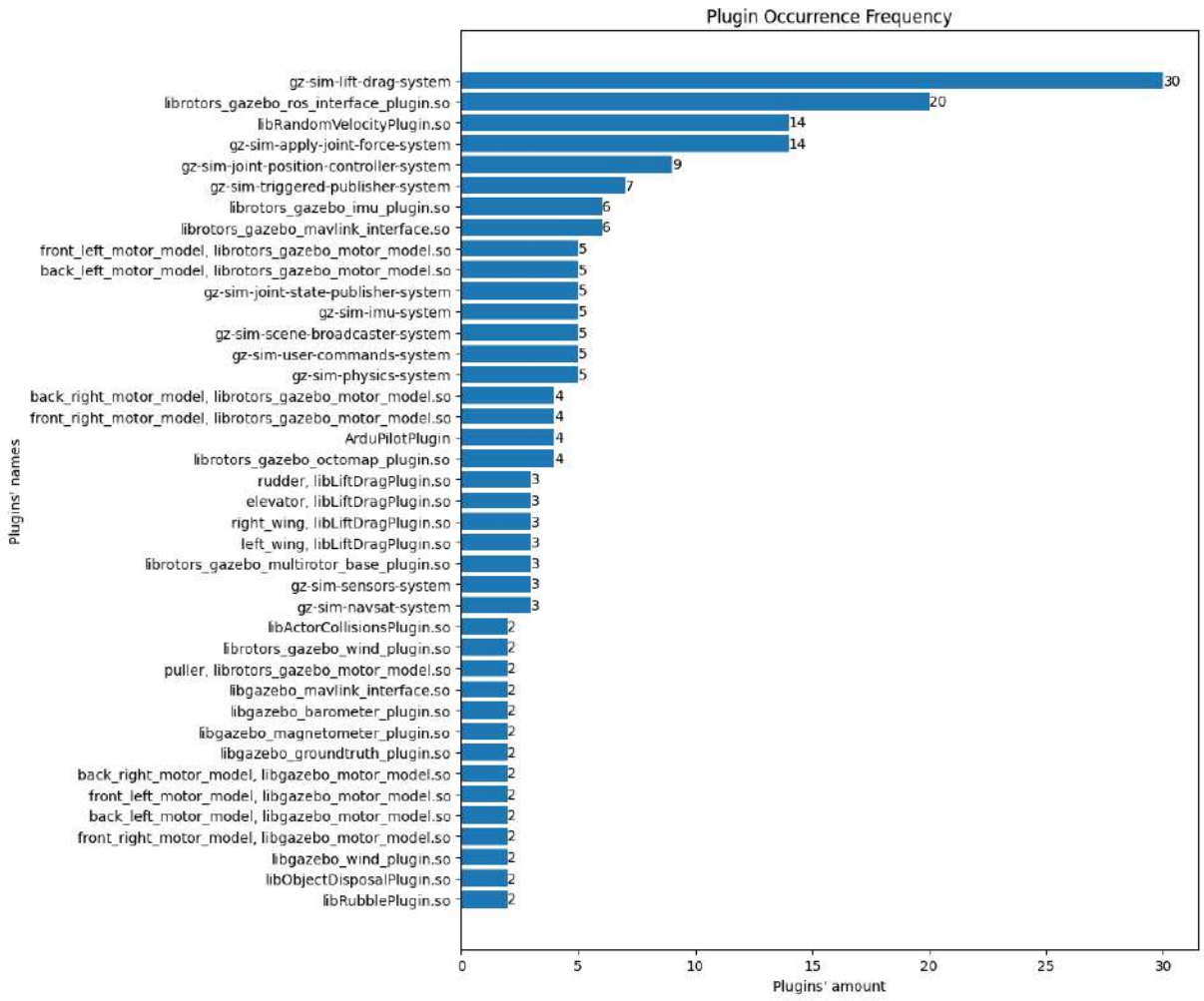

Figure 5.5: Plugin Types Occurrence Frequency in Worlds

Even more than in the previous section with plugins, there are worlds without <joint> elements. 144 out of 164 worlds (88%) include no joints. Since joints are used to connect links of a model (rigid parts) allowing relative motion between them, the absence of joints indicates that this world has no moving robots. In conclusion, the overwhelming majority of worlds are presumably not dynamic.

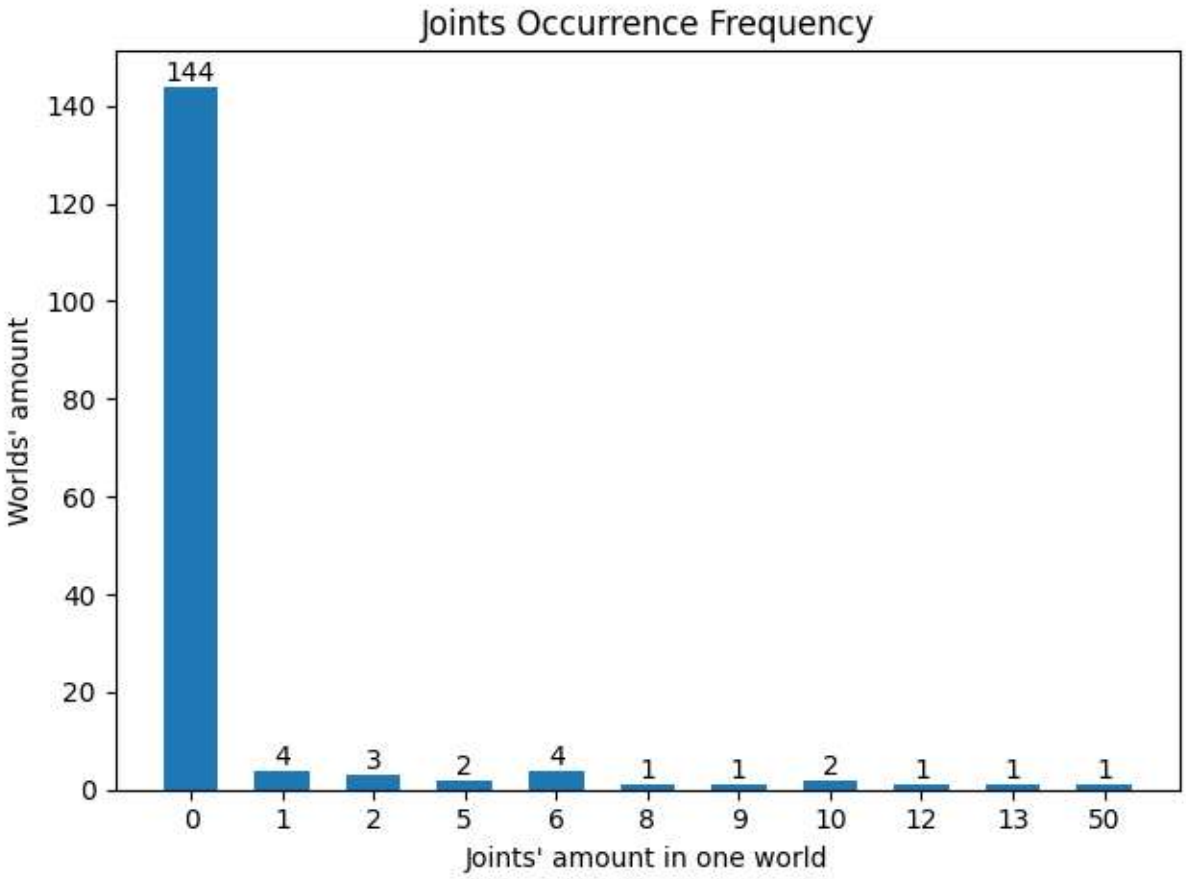

Figure 5.6: Joints Occurrence Frequency in Worlds

Sensors and actors are also rarely met in worlds from the dataset. Only 11 worlds (6%) have sensors (Figure 5.7) in them and only 2 worlds (1%) have actors in them.

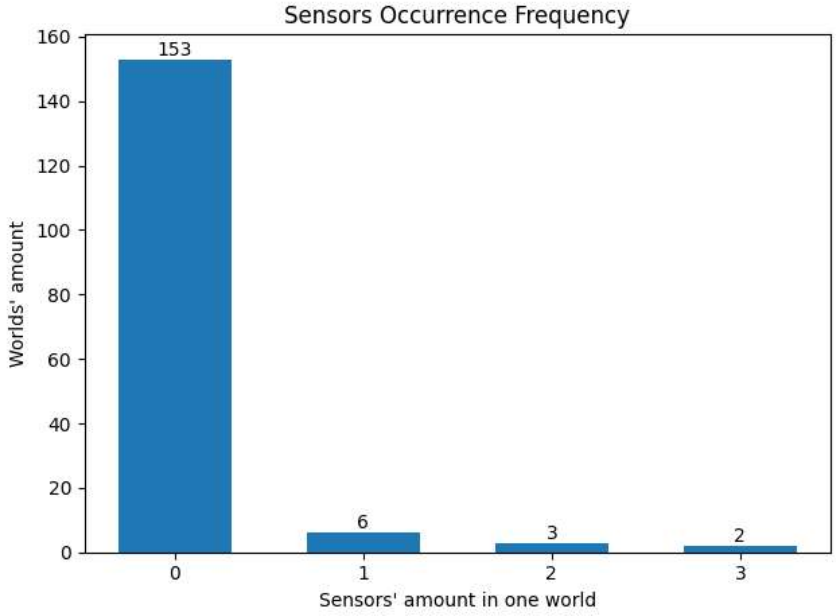

Figure 5.7: Sensors Occurrence Frequency in Worlds

A link is an important part of a model. Counting links gives one more additional look on the complexity of the worlds. The situation is quite similar to the one with <model> and <include> elements. Mostly, from 0 to 2 links are inside one Gazebo world.

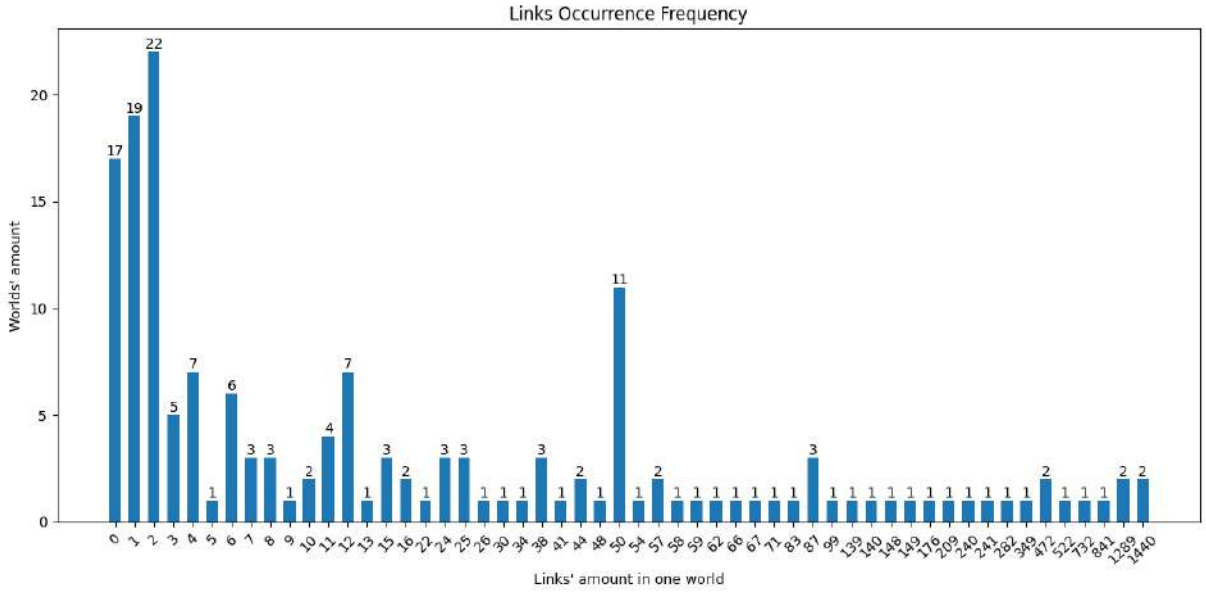

Figure 5.8: Links Occurrence Frequency in Worlds

Plugins, joints, and sensors fall into one category, so they were also calculated at the end together. 143 out of 164 worlds (87%) of worlds have no plugins, joints, and sensors in them. On Figure 5.9, a distribution of the rest of the worlds might be seen.

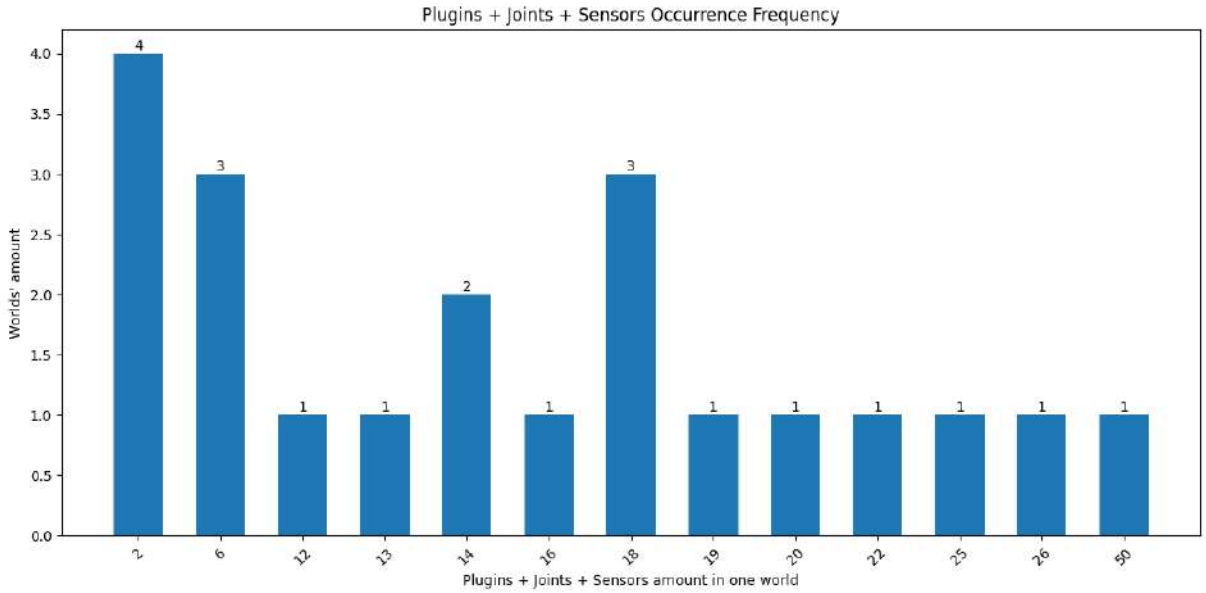

Figure 5.9: Plugins, Joints, and Sensors Occurrence Frequency in Worlds

5.3 Gazebo Models Analysis

Models analysis was started with nested <model> elements. It is rare to face such a model. Only 6 models have nested models, namely 'office_small', 'iris_hitl', 'iris_triple_depth_camera', 'exp_empty_rooms', 'conveyor_belt_tall_ariac', and 'conveyor_belt_ariac'. These models are rather complex.

Next, <include> elements were taken into consideration. <include> elements start with "model://" but there are nine <include> elements where this pattern was not met, even though they represented models as well. All the <include> elements are direct children of models, except for one.

94% of model files do not have <include> elements. Consequently, it is regarded as a rare practice and most models are not very complex.

Among the models containing <include> elements, a few featured a large number of included models, with a maximum of 28 included models. 19 models have only one <include>, while 26 models have two.

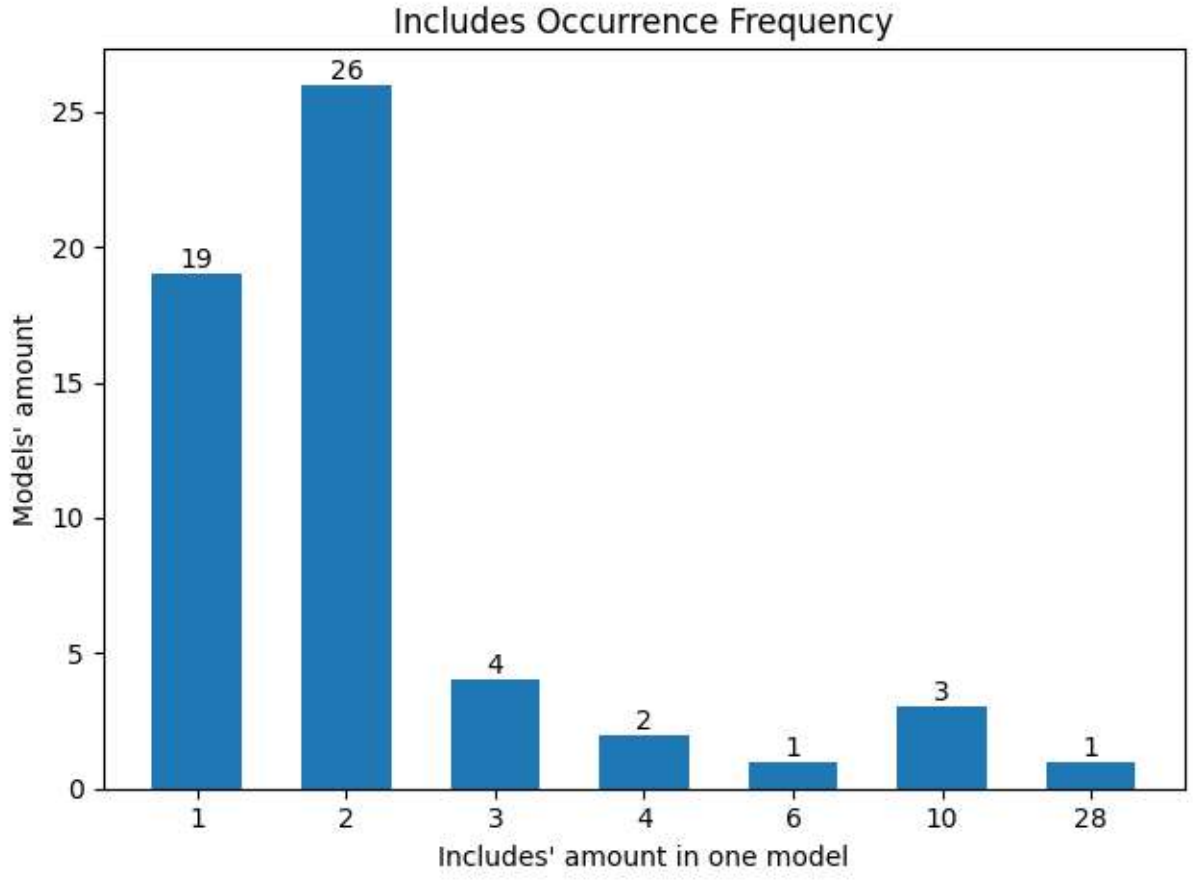

Figure 5.10: Includes Occurrence Frequency in Models

968 out of 1029 models (94%) have zero <model> and <include> elements meaning that in most cases models are created independently, without the need to separate logic

in separate files or models. Only one model, specifically iris_triple_depth_camera has both <model> and <include> elements. The total number of them is 5.

As the next step, plugins in models were counted. 90% of the models do not contain plugins. In Figure 5.11, the rest of the models that have one or more plugins are taken into account. A decent amount of considered models (43%) include only one plugin.

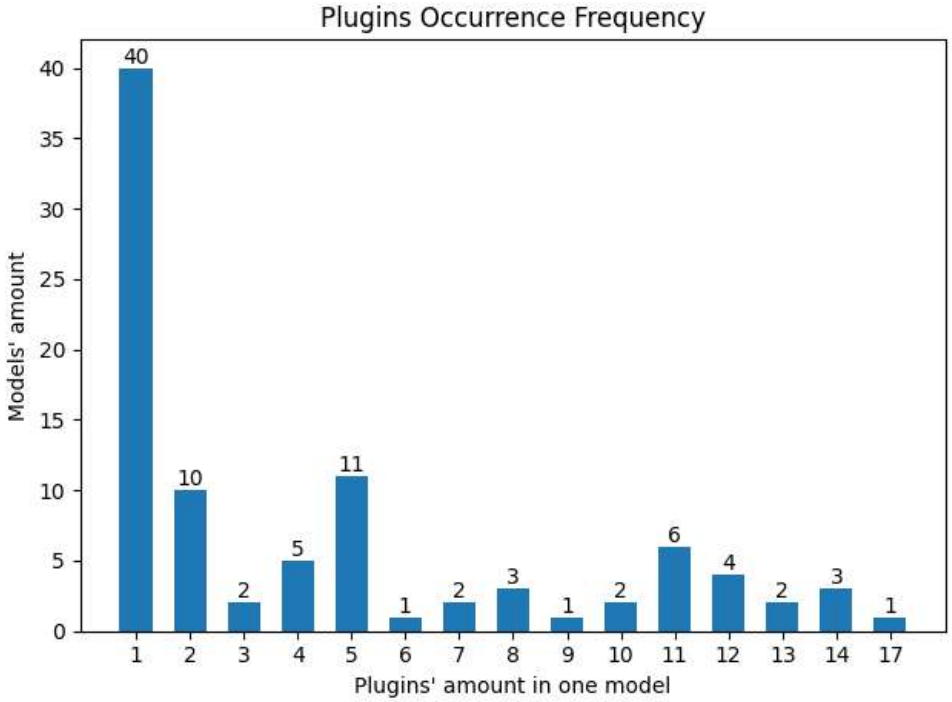

Figure 5.11: Plugins Occurrence Frequency in Models

It follows from Figure 5.12 that *gz-sim-lift-drag-system* is the most widely used in the models from the dataset. Plugins that occurred less than 4 times were not included in the plot.

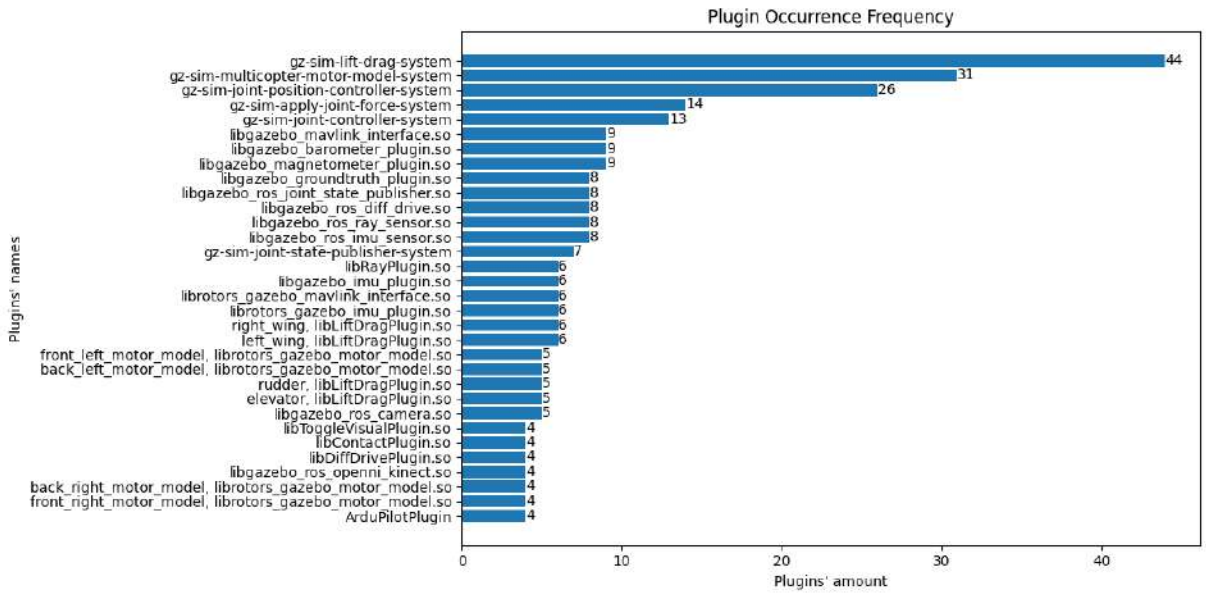

Figure 5.12: Plugin Types Occurrence Frequency in Models

What is the difference between plugins count in worlds and in models? (maybe not only plugins)

791 out of 1029 models (76%) are static (include <static> element in them). If static is set to true, the model is immovable; i.e. a dynamics engine will not update its position. [47]

914 out of 1029 models (88%) do not contain any joints, which also leads to a conclusion about immobility of the vast majority of models. The same pattern, where the amount of plugins is oppositely proportional to the amount of worlds with such an amount of plugins, is observed with joints.

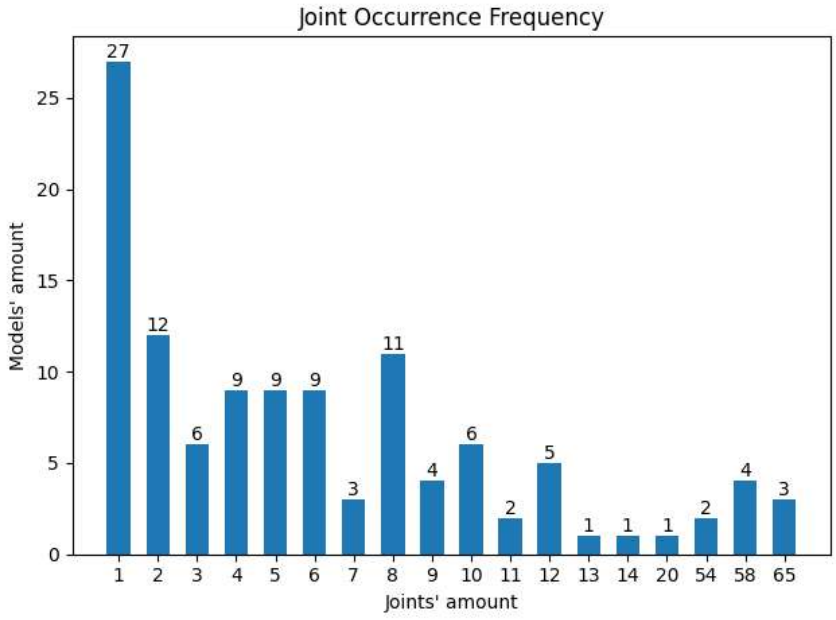

Figure 5.13: Joints Occurrence Frequency in Models

In the next phase, sensors were analyzed. They also occur rarely in models in the dataset. 92% of the models have zero sensors. In those that contain at least one sensor, 53% include only one sensor.

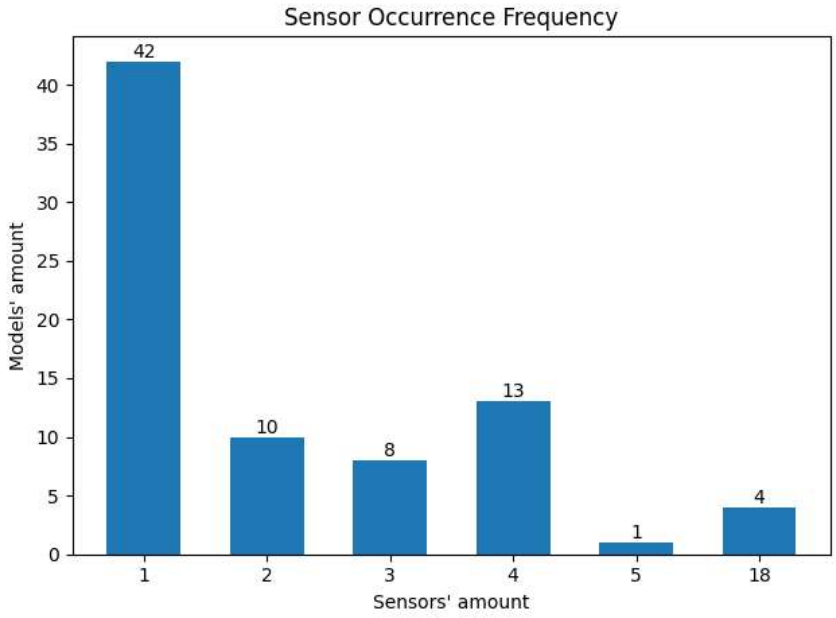

Figure 5.14: Sensors Occurrence Frequency in Models

Finally, the ranking of sensor types was established (Figure 5.15). Camera sensor is the most frequently used. There are a few others that are also related to camera (e.g., *wide_stereo_gazebo_r_stereo_camera_sensor*).

Another category of sensor is devoted to defining robot's orientation and velocity. *imu_sensor*, *navsat_sensor*, *laser*, *gps* fall into this category.

In addition, contact sensors are also present in the list (e.g., *r_gripper_r_finger_tip_contact_sensor*, *torso_lift_contact_sensor*), although they are less common than other types.

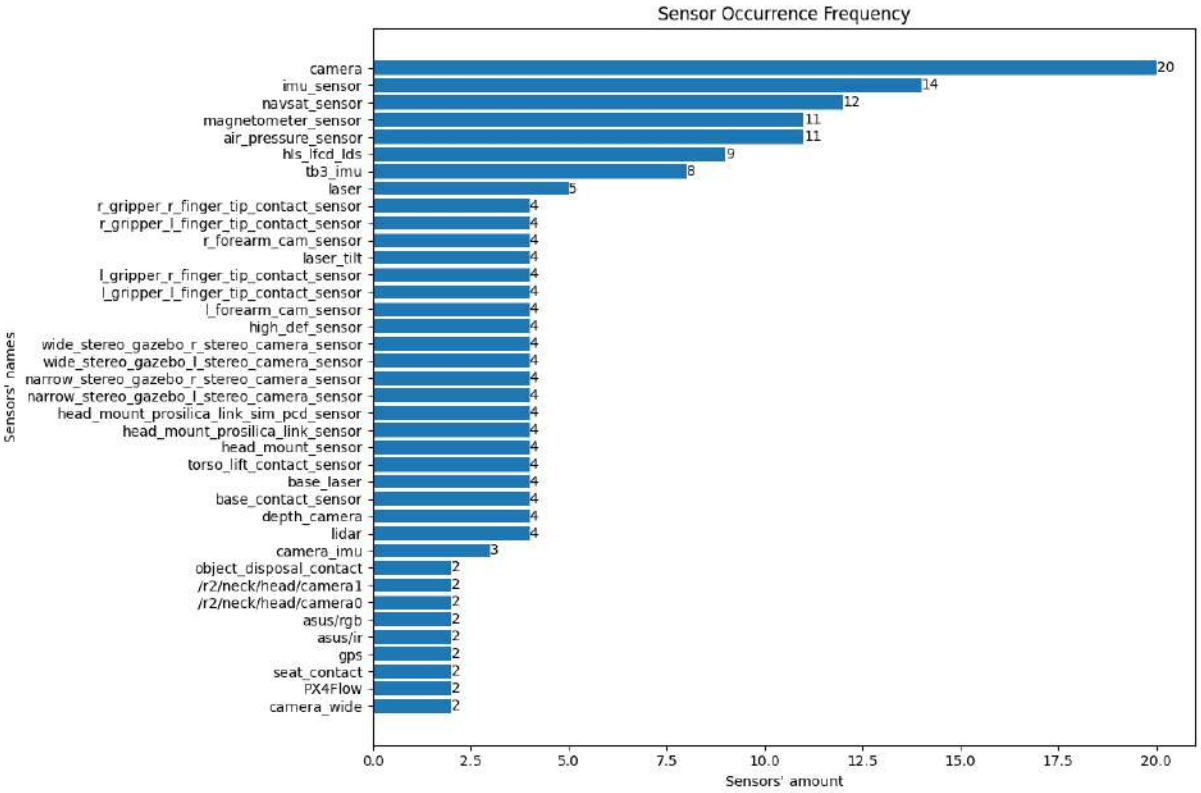

Figure 5.15: Sensor Types Occurrence Frequency in Models

5.4 Reference to Analysis

The implementation of the code used to obtain the statistics and graphs seen below can be found here https://github.com/koejdga/gazebo_worlds_analysis.

5.5 Interpretation of Results

Element	0	1-5	6+	Max amount
<model>	42%	21%	37%	658
<include>	36%	32%	32%	541
<model> and <include>	0%	41%	59%	1080
<plugin>	74%	18%	8%	49
<joint>	88%	5%	7%	50
<sensor>	93%	7%	0%	3
<link>	10%	33%	57%	1440
<plugin>, <joint>, and <sensor>	87%	2%	11%	50

Table 5.2: Conclusions About **Gazebo Worlds** Analysis

Since the vast majority of them did not include moving objects in them, it could be useful to try and fill in this gap, specifically to create a Gazebo world with a moving object.

Element	0	1-5	6+	Max amount
<include>	94%	5%	1%	28
<plugin>	90%	7%	3%	17
<joint>	88%	6%	6%	65
<sensor>	92%	7%	1%	18

Table 5.3: Conclusions About **Gazebo Models** Analysis

What would a world with a moving object and a UAV be useful for? Such an environment enables an interaction between a drone and a moving object. The algorithms designed for tracking purposes may be tested in the mentioned environment. These algorithms have a variety of nuances. Firstly, it makes sense to test how good the algorithm copes with the tracking with different speeds of a moving object. High speed usually presents more difficulties. UAV sensors have a finite frame rate or sampling rate, so they may not capture enough data points quickly enough. Also, high-speed objects often cause motion blur in visual sensors, degrading image quality and making it harder for computer vision algorithms to detect and identify the object precisely. In addition, there are different types of tracking. The first one is visual tracking, where the drone does not change its own position but rather hovers in one place and processes camera feed to detect moving targets. Another one is target following, where the drone adjusts its position to maintain the target within its field of view. In the latter case limits of a UAV maneuverability also play a role in the algorithm's effectiveness when an object moves at high speed. Furthermore, the process of following in a real-life environment is usually accompanied with obstacle avoidance and thorough path planning. Based on the foregoing, it follows that the object tracking algorithms require scrupulous testing. The Gazebo world with a moving object has potential to be used further to meet these needs.

Commonly, the tracked object is a type of vehicle. Therefore, the Gazebo simulation environment was designed to include a car model that closely resembles a real-world automobile.

Chapter 6

Development of a UAV Testing Environment with a Moving Object

6.1 Solution

According to the Gazebo website [26], Gazebo Jetty, Gazebo Harmonic, and Gazebo Fortress are long-term support versions of Gazebo. At the same time, Gazebo Harmonic is a version recommended for use on Ubuntu 24.04 Noble, Ubuntu 22.04 Jammy, Mac Ventura, and Mac Monterey [64]. Based on this, Gazebo Harmonic appeared to be the most appropriate candidate for the development and launching of a new Gazebo world.

As it was mentioned in the previous section, an automobile was chosen as the moving object. The visual appearance of the car might also differ depending on circumstances. For instance, some tracking algorithms might be trained specifically on military vehicles. In turn, other ones are aimed more at civilian means of transport.

In this thesis, a light-weight asset with a red passenger car was utilized. It is considered a good starting point and can be changed without complications by adding the desired model into the project.

The purpose is to make the automobile move in a circle. Such a setting would allow developers to test UAV algorithms continuously, by making changes or tweaks to them, and testing again, without the necessity to restart the Gazebo environment every time because the car's trajectory repeats itself perpetually, so the moving object is not gone away from the starting point after the time passed. It just moves in a circle of a medium radius.

The base for this world was taken from `https://github.com/ArduPilot/ardupilot_gazebo`, which is the official ArduPilot plugin for Gazebo that provides support for the recent releases of the Gazebo simulator (Gazebo Garden), (Gazebo Harmonic) and (Gazebo Ionic). The base for the created world is named "iris_runway.sdf". It contains Iris quadcopter model with a camera on a gimbal (a device that helps keep the camera stable even

while the drone is flying at fast speeds, which enhances the quality of the images and particularly the video).

Also, "iris_runway.sdf" world contains an infinite plane that looks like a runway and is surrounded by grass.

The car model was downloaded from the Internet. It consists of one .dae file. Based on the COLLADA format, a file with the DAE (Digital Asset Exchange) file extension is a digital asset exchange file. As the name implies, this XML-based format is used by various graphics programs to exchange digital assets like images, textures, and 3D models. [65]

A good place to find a suitable model is https://app.gazebosim.org/fuel/models. Gazebo Fuel hosts hundreds of models that can easily be added to a world running in the Gazebo GUI. [66] Among the available models, there are aerial and ground vehicles, different kinds of terrains, people, and non-moving objects, e.g., furniture, household items, dinnerware etc. Additionally, one may find an appropriate world for the project there.

To make the automobile move, two links, left_wheel and right_wheel, and also a link for caster wheel, were added. To connect them to the main body, three joints were included as well. These wheels serve the purpose of enabling motion and provide practical functionality in the simulation. They are not visually represented in the world and do not serve any visual function.

Finally, gz::sim::systems::DiffDrive plugin is used to make the car move. This is a differential drive controller which can be attached to a model with any number of left and right wheels. In Listing 6.1, it can be seen how left and right joints were declared inside it. They control the left and right wheel correspondingly. ¡wheel_separation¿ sets the distance between wheels, in meters. ¡wheel_radius¿ sets a wheel radius in meters. ¡odom_publish_frequency¿ set an odometry publication frequency. ¡topic¿ sets a custom topic that this system will subscribe to in order to receive command velocity messages. [67]

```
<!--diff drive plugin-->
<plugin filename="gz-sim-diff-drive-system" name="
    gz::sim::systems::DiffDrive">
     <left_joint>left_wheel_joint</left_joint>
     <right_joint>right_wheel_joint</right_joint>
     <wheel_separation>1.2</wheel_separation>
     <wheel_radius>0.4</wheel_radius>
     <odom_publish_frequency>1</odom_publish_frequency>
     <topic>cmd_vel</topic>
</plugin>
```

Listing 6.1: DiffDrive plugin in the world

In order to simplify the movement triggering, KeyPublisher and TriggeredPublisher were added.

The TriggeredPublisher system publishes a user specified message on an output topic in response to an input message that matches user specified criteria. [68] This means that after pressing the R key a message to the cmd_vel topic is sent and processed by the car model. This results in a car moving in a circle.

```
    <!-- Start Movement -->
    <plugin filename="gz-sim-triggered-publisher-system" name="
   gz::sim::systems::TriggeredPublisher">
      <input type="gz.msgs.Int32" topic="/keyboard/keypress">
        <match field="data">82</match>        <!-- press R -->
      </input>
      <output type="gz.msgs.Twist" topic="/cmd_vel">
                linear: {x: 2}, angular: {z: 0.4}
      </output>
    </plugin>

    <!-- Stop Movement -->
    <plugin filename="gz-sim-triggered-publisher-system" name="
   gz::sim::systems::TriggeredPublisher">
      <input type="gz.msgs.Int32" topic="/keyboard/keypress">
        <match field="data">83</match>        <!-- press S -->
      </input>
      <output type="gz.msgs.Twist" topic="/cmd_vel">
                linear: {x: 0}, angular: {z: 0}
      </output>
    </plugin>
```

Listing 6.2: TriggeredPublisher plugin in the world

To enable a speed change of the car, a bash script (Listing 6.3) was added. It takes one number (an integer or a double) as an input and uses it to send a command to the cmd_vel topic. This leads to the car slowing down or speeding up.

```
#!/bin/bash

# check if the first argument is either an integer or a double
if [[ $1 =~ ^-?[0-9]+$ ]] || [[ $1 =~ ^-?[0-9]*\.[0-9]+$ ]]; then
  angular_value=$(echo "$1 / 5" | bc -l)
  gz topic -t "/cmd_vel" -m gz.msgs.Twist -p "linear: {x: $1}, angular:
   {z: $angular_value}"
else
  echo "Usage: ./move_in_circle.sh <number>, where <number> is speed,
   standard = 2"
fi
```

Listing 6.3: Script for altering the speed of the car

This world should be run together with ArduPilot, since the drone with ArduPilot software is integrated in this world. Additionally, the the the ground control station may be connected to the running ArduPilot to control through arough arough a graphical interface.

To find out more about moving robots in Gazebo, use this link `https://gazebosim.org/docs/latest/moving_robot/`.

6.2 Demonstration

In Figure 6.1, the car and the drone are in their original positions. Whereas in Figure 6.2, the car moved and the drone is flying in the air.

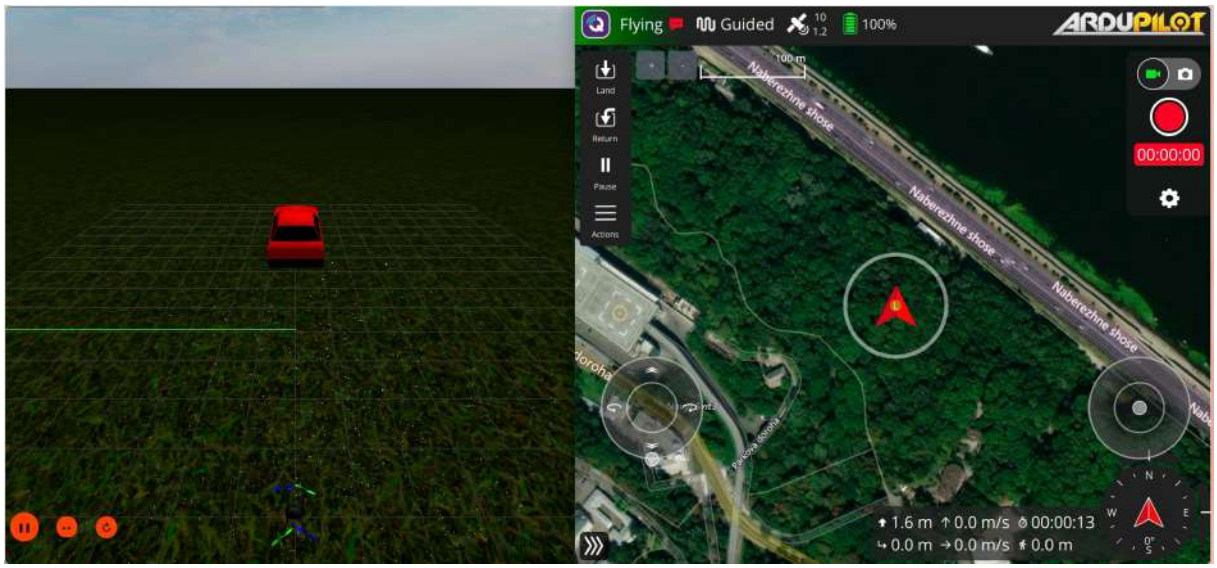

Figure 6.1: Sensor Types Occurrence Frequency in Models

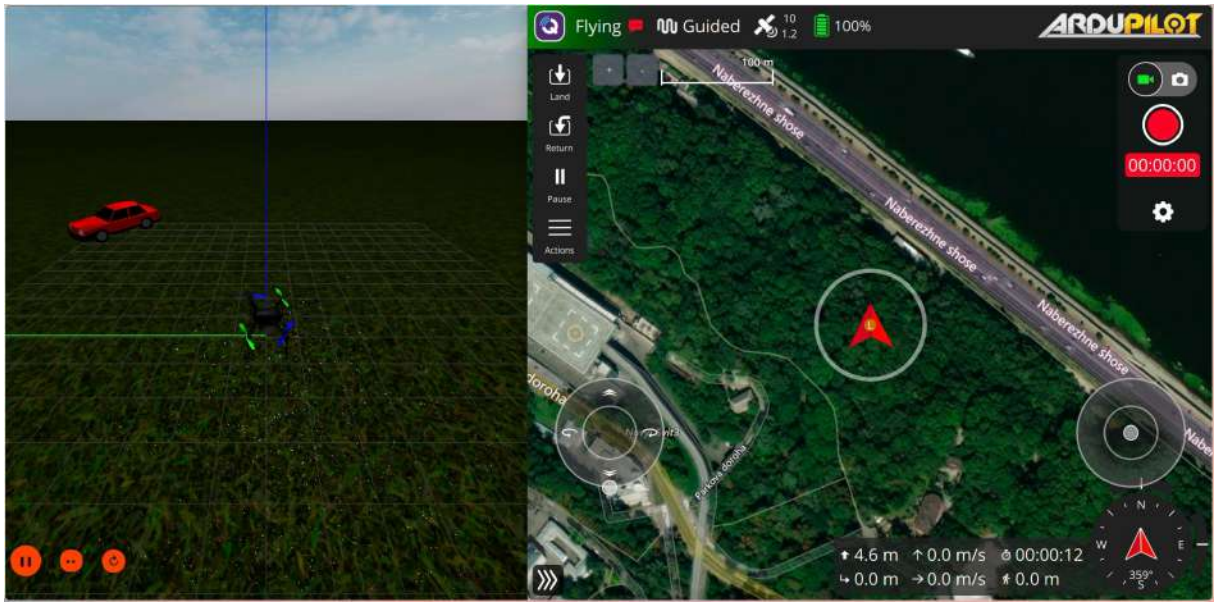

Figure 6.2: Sensor Types Occurrence Frequency in Models

Chapter 7

Conclusion and Future Work

This paper offers an overview of the four prominent robotics simulators in the context of UAVs. It introduces the statistics based on two scientific search engines, Google Scholar and Semantic Scholar, which provides a way to analyze the mentioned UAV simulators, as well as the flight control software. The most popular and robust simulator is Gazebo, Microsoft AirSim is already deprecated, BetaFlight is developing, and NVIDIA Isaac Sim is developing as well aiming at reinforcement learning in robotics. Over the years, all the simulators and software acquired more popularity, which shows that the aerial robotics sphere continues to develop actively while having lots of obstacles to face and solve yet. Also, it was observed that Gazebo with PX4 and ArduPilot are yet two most common pairings of a simulator and flight control software.

After looking at the statistics, a deeper look into Gazebo worlds available on the Internet was taken and a list of them was composed and analyzed. Finally, a new Gazebo world was implemented, which includes a moving automobile and a UAV. The car moves in a circle, can be started and stopped via keyboard controls, the speed of the car might be adjusted through the developed script. The UAV can be connected to ArduPilot SITL using Gazebo ArduPilot plugin. The created setup could be useful for the object tracking algorithms' testing or other conditions where interaction between a drone and a moving object is necessary. Currently, no evaluations of object tracking algorithms were performed with the created world. The future work might be devoted to this. Additionally, this world may be used as a basis to more complex worlds, e.g., with generated obstacles, several vehicles etc. Finally, it is possible to gather new statistics utilizing the scripts provided in this thesis. This might reveal new insights in the field of UAV simulators.

Bibliography

[1] James Patton Rogers, ed. *De Gruyter Handbook of Drone Warfare*. 1st. De Gruyter Contemporary Social Sciences Handbooks 4. Boston: De Gruyter, 2024. ISBN: 978-3-11-074192-6.

[2] Honghao Pan et al. "T-STAR: Time-Optimal Swarm Trajectory Planning for Quadrotor Unmanned Aerial Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* (2025), pp. 1–16. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2025.3557783. (Visited on 05/14/2025).

[3] Drew Hanover et al. "Autonomous Drone Racing: A Survey". In: *IEEE Transactions on Robotics* 40 (2024). Comment: 26 pages, pp. 3044–3067. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2024.3400838. arXiv: 2301.01755 [cs]. (Visited on 04/23/2025).

[4] *Iris+ Is the First Consumer Drone to Be Able to Follow Its User*. https://newatlas.com/3d-robotics-iris-plus-follow-me-drone/33747/. Accessed: 2025-05-21.

[5] Tyler Gregory Hicks, S. David Hicks, and Joseph Leto, eds. *Standard Handbook of Engineering Calculations*. 3rd ed. Includes index. New York: McGraw-Hill, 1995. ISBN: 978-0-07-028812-6.

[6] F. Nex et al. "UAV in the Advent of the Twenties: Where We Stand and What Is Next". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 184 (Feb. 2022), pp. 215–242. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2021.12.006. (Visited on 05/02/2025).

[7] Xudong Luo, Yiquan Wu, and Langyue Zhao. "YOLOD: A Target Detection Method for UAV Aerial Imagery". In: *Remote Sensing* 14.14 (July 2022), p. 3240. ISSN: 2072-4292. DOI: 10.3390/rs14143240. (Visited on 05/14/2025).

[8] Shubhani Aggarwal and Neeraj Kumar. "Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges". In: *Computer Communications* 149 (Jan. 2020), pp. 270–299. ISSN: 01403664. DOI: 10.1016/j.comcom.2019.10.014. (Visited on 05/14/2025).

[9] Enrique Aldao et al. “UAV Obstacle Avoidance Algorithm to Navigate in Dynamic Building Environments”. In: *Drones* 6.1 (Jan. 2022), p. 16. ISSN: 2504-446X. DOI: 10.3390/drones6010016. (Visited on 05/14/2025).

[10] Haojia Li et al. “AutoTrans: A Complete Planning and Control Framework for Autonomous UAV Payload Transportation”. In: *IEEE Robotics and Automation Letters* 8.10 (Oct. 2023), pp. 6859–6866. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2023.3313010. (Visited on 05/14/2025).

[11] Yongkun Zhou, Bin Rao, and Wei Wang. “UAV Swarm Intelligence: Recent Advances and Future Trends”. In: *IEEE Access* 8 (2020), pp. 183856–183878. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3028865. (Visited on 05/14/2025).

[12] Franc Mihalič, Mitja Truntič, and Alenka Hren. “Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges”. In: *Electronics* 11.15 (Aug. 2022), p. 2462. ISSN: 2079-9292. DOI: 10.3390/electronics11152462. (Visited on 05/14/2025).

[13] *Hardware in the Loop Simulation (HITL)*. https://docs.px4.io/v1.12/en/simulation/hitl.html. Accessed: 2025-05-21.

[14] *Simulation-In-Hardware (SIH)*. http://docs.px4.io/main/en/sim_sih/. Accessed: 2025-05-20.

[15] Janis Peksa and Dmytro Mamchur. “A Review on the State of the Art in Copter Drones and Flight Control Systems”. In: *Sensors* 24.11 (May 2024), p. 3349. ISSN: 1424-8220. DOI: 10.3390/s24113349. (Visited on 05/29/2025).

[16] Aicha Idriss Hentati et al. “Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis”. In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. Limassol: IEEE, June 2018, pp. 1495–1500. ISBN: 978-1-5386-2070-0. DOI: 10.1109/IWCMC.2018.8450505. (Visited on 05/02/2025).

[17] Jack Collins et al. “A Review of Physics Simulators for Robotic Applications”. In: *IEEE Access* 9 (2021), pp. 51416–51431. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3068769. (Visited on 05/02/2025).

[18] Cora A Dimmig et al. “Survey of Simulators for Aerial Robots”. In: ().

[19] Emad Ebeid et al. “A Survey of Open-Source UAV Flight Controllers and Flight Simulators”. In: *Microprocessors and Microsystems* 61 (Sept. 2018), pp. 11–20. ISSN: 01419331. DOI: 10.1016/j.micpro.2018.05.002. (Visited on 05/29/2025).

[20] *Ukrainian Fight Drone Simulator*. https://store.steampowered.com/app/2862860/Ukrainian_Fight_Drone_Simulator/. Accessed: 2025-05-20.

[21] Abhishek Phadke et al. “Designing UAV Swarm Experiments: A Simulator Selection and Experiment Design Process”. In: *Sensors* 23.17 (Aug. 2023), p. 7359. ISSN: 1424-8220. DOI: 10.3390/s23177359. (Visited on 05/29/2025).

[22] *Gazebo*. https://github.com/gazebosim. Accessed: 2025-05-28.

[23] *About Gazebo*. https://gazebosim.org/about. Accessed: 2025-05-02.

[24] N. Koenig and A. Howard. “Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. Sendai, Japan: IEEE, 2004, pp. 2149–2154. ISBN: 978-0-7803-8463-7. DOI: 10.1109/IROS.2004.1389727. (Visited on 05/03/2025).

[25] Morgan Quigley et al. “ROS: An Open-Source Robot Operating System”. In: ().

[26] *Gazebo Releases*. https://gazebosim.org/docs/latest/releases/. Accessed: 2025-05-02.

[27] *Feature Comparison*. https://gazebosim.org/docs/latest/comparison/. Accessed: 2025-05-03.

[28] *ROS with Gazebo Classic Simulation*. https://docs.px4.io/main/en/simulation/ros_interface.html. Accessed: 2025-05-21.

[29] *Beginner: Overview*. https://classic.gazebosim.org/tutorials?tut=guided_b1. Accessed: 2025-05-03.

[30] *Gazebo Simulation Environment Requirements and Limitations*. https://www.mathworks.com/help/robotics/ug/gazebo-simulation-requirements.html. Accessed: 2025-05-03.

[31] Yunlong Song et al. “Flightmare: A Flexible Quadrotor Simulator”. In: ().

[32] *Flightmare*. https://github.com/uzh-rpg/flightmare. Accessed: 2025-05-03.

[33] *AirSim: A Simulator to Help AI Research for Use in Drones*. https://www.commercialuavnews.com/public-safety/airsim-simulator-help-artificial-intelligence-research-use-drones. Accessed: 2025-05-28.

[34] Shital Shah et al. *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*. Comment: Accepted for Field and Service Robotics conference 2017 (FSR 2017). July 2017. DOI: 10.48550/arXiv.1705.05065. arXiv: 1705.05065 [cs]. (Visited on 05/03/2025).

[35] *NVIDIA Isaac Sim*. https://developer.nvidia.com/isaac/sim. Accessed: 2025-05-21.

[36] Marcelo Jacinto et al. *Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation*. Comment: This paper has been accepted for publication in the 2024 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE. Apr. 2024. DOI: 10.48550/arXiv.2307.05263. arXiv: 2307.05263 [cs]. (Visited on 05/28/2025).

[37] *NVIDIA Isaac Lab*. https://developer.nvidia.com/isaac/lab. Accessed: 2025-05-21.

[38] Rita Vine. "Google Scholar". In: *Journal of the Medical Library Association* 94.1 (Jan. 2006), pp. 97–99. ISSN: 1536-5050. (Visited on 05/02/2025).

[39] Suzanne Fricke. "Semantic Scholar". In: *Journal of the Medical Library Association* 106.1 (Jan. 2018). ISSN: 1558-9439, 1536-5050. DOI: 10.5195/jmla.2018.280. (Visited on 05/02/2025).

[40] *About Semantic Scholar*. https://www.semanticscholar.org/about. Accessed: 2025-05-02.

[41] *PX4-Autopilot*. https://github.com/PX4/PX4-Autopilot. Accessed: 2025-05-22. (Visited on 05/22/2025).

[42] *Ardupilot*. https://github.com/ArduPilot/ardupilot. Accessed: 2025-05-22.

[43] *Inav*. https://github.com/iNavFlight/inav. Accessed: 2025-05-22.

[44] *Betaflight*. https://github.com/betaflight/betaflight. Accessed: 2025-05-22.

[45] *AirSim Announcement: This Repository Will Be Archived in the Coming Year*. https://microsoft.github.io/AirSim/. Accessed: 2025-05-22.

[46] *Building a World*. https://classic.gazebosim.org/tutorials?tut=build_world. Accessed: 2025-05-16.

[47] *SDFormat*. http://sdformat.org/spec?ver=1.12. Accessed: 2025-05-19.

[48] *Gazebo Models and Worlds Collection*. https://github.com/leonhartyao/gazebo_models_worlds_collection. Accessed: 2025-05-21.

[49] *3DGEMS*. https://data.nvision2.eecs.yorku.ca/3DGEMS/. Accessed: 2025-05-21.

[50] *RotorS*. https://github.com/ethz-asl/rotors_simulator/tree/master. Accessed: 2025-05-21.

[51] *Gazebo Model Collection*. https://github.com/tudelft/gazebo_models/tree/master. Accessed: 2025-05-21.

[52] *Gazebo Worlds*. https://github.com/ARTI-Robots/gazebo_worlds/tree/master. Accessed: 2025-05-21.

[53] *Clearpath Additional Simulation Worlds*. https://github.com/clearpathrobotics/cpr_gazebo/tree/noetic-devel. Accessed: 2025-05-21.

[54] *Fetch Gazebo*. https://github.com/ZebraDevs/fetch_gazebo/tree/gazebo9. Accessed: 2025-05-21. (Visited on 05/21/2025).

[55] *Dataset-of-Gazebo-Worlds-Models-and-Maps*. https://github.com/mlherd/Dataset-of-Gazebo-Worlds-Models-and-Maps. Accessed: 2025-05-21.

[56] *Useful World Files for Gazebo and ROS 2 Simulations*. https://store.steampowered.com/app/2862860/Ukrainian_Fight_Drone_Simulator/. Accessed: 2025-05-21.

[57] *Gazebo Worlds*. https://docs.px4.io/main/en/sim_gazebo_gz/worlds.html. Accessed: 2025-05-21.

[58] *Gazebo Models and Worlds*. https://github.com/eliabntt/gazebo_resources. Accessed: 2025-05-21.

[59] *Gazebo Models and Worlds*. https://eliabntt.github.io/gazebo_resources/. Accessed: 2025-05-21.

[60] *Aws-Robotics*. https://github.com/orgs/aws-robotics/repositories?q=gazebo. Accessed: 2025-05-21.

[61] *ArduPilot Gazebo Plugin*. https://github.com/ArduPilot/ardupilot_gazebo. Accessed: 2025-05-21.

[62] *Clearpath Robotics*. https://clearpathrobotics.com. Accessed: 2025-05-19.

[63] Fadri Furrer et al. "RotorS—A Modular Gazebo MAV Simulator Framework". In: *Robot Operating System (ROS)*. Ed. by Anis Koubaa. Vol. 625. Cham: Springer International Publishing, 2016, pp. 595–625. ISBN: 978-3-319-26052-5 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. (Visited on 05/22/2025).

[64] *Getting Started with Gazebo?* https://gazebosim.org/docs/latest/getstarted/. Accessed: 2025-05-02.

[65] *DAE File: What It Is & How to Open One*. https://www.lifewire.com/dae-file-2620544. Accessed: 2025-05-16.

[66] *Model Insertion from Fuel*. https://gazebosim.org/docs/latest/fuel_insert/. Accessed: 2025-05-16.

[67] *DiffDrive Class Reference*. https://gazebosim.org/api/sim/8/classgz_1_1sim_1_1systems_1_1DiffDrive.html#details. Accessed: 2025-05-20.

[68] *Triggered Publisher*. https://gazebosim.org/api/sim/7/triggeredpublisher.html. Accessed: 2025-05-20.