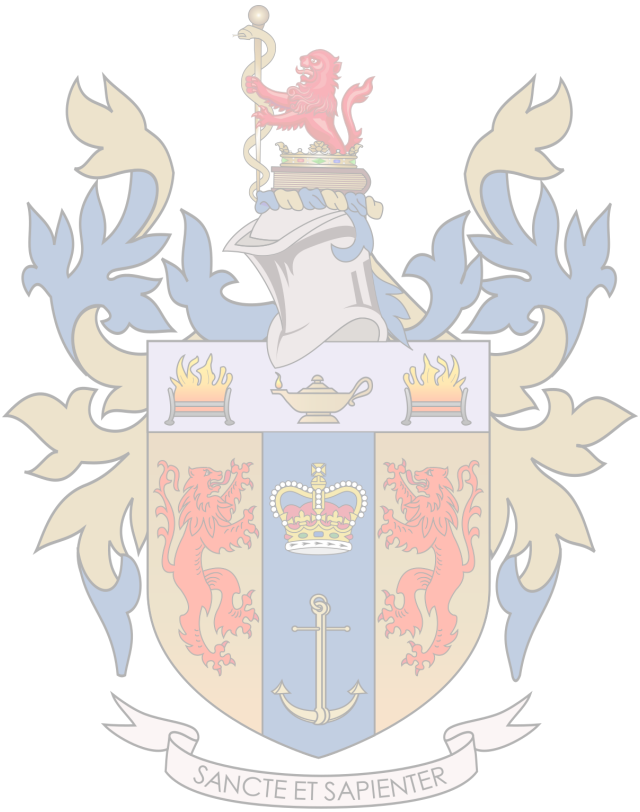
Sim-to-Real Framework for Autonomous Drone-Based System in Warehouses

William Droin, Student Number: K23115232

King's College London, Department of Natural & Mathematical Sciences
MSc in Robotics
15th of August 2024



Abstract

This paper explores the development of a Sim-to-Real framework aimed at facilitating the deployment of autonomous drone-based systems for conducting safety audits in warehouse environments. Leveraging Nvidia's Omniverse and Isaac Sim, a highly detailed virtual warehouse was constructed, serving as a simulation environment for drone navigation and object detection tasks. The system integrates ROS and Mavlink protocols to ensure seamless communication between drones and external control systems. This framework enables researchers to develop and test safety monitoring strategies in a controlled, cost-effective simulation environment before deploying them in real-world settings. The implementation focuses on detecting workers' compliance with safety regulations by identifying the presence of safety vests. The experimental results indicate promising performance in simulation, though challenges remain in achieving consistent accuracy when applied to real-world data. The framework's flexibility and modularity pave the way for further advancements in drone-based indoor monitoring systems, with potential applications extending beyond safety audits to include inventory management and surveillance tasks. Future work will address the limitations identified in real-world deployment, particularly the need for more diverse and representative training datasets to improve model generalisation.

Contents

1 Nomenclature

The following is a list of definitions for terms used throughout this project:

- **Sim-to-Real**: The process of using virtual simulations to develop and test robots before using them in the real world. It allows developers to safely and cheaply try out their designs in a computer-generated environment that mimics real-life conditions. Once the robot performs well in the simulation, it is then used in the real world, reducing risks and ensuring it works as expected.
- **Reinforcement learning**: A type of machine learning where an algorithm learns to make decisions by interacting with its environment and receiving feedback. This process is akin to training a dog: when a dog performs a trick correctly, it receives a reward (like a treat), and when it performs incorrectly, it receives no reward. Over time, the dog learns which actions lead to positive outcomes. Similarly, in reinforcement learning, the algorithm takes various actions, receives rewards or penalties based on the outcomes, and uses this feedback to learn the best actions to maximise cumulative rewards. This trial-and-error method helps the algorithm improve its performance in achieving specific goals.
- **ROS**: A framework that provides tools and libraries for building and controlling robots. It acts like an operating system for robots, enabling different software components to work together seamlessly. Just as Windows or Linux manage programs on a computer, ROS manages the communication between a robot's sensors, controllers, and actuators, making it easier for developers to create and integrate robotic applications.
- **ROS topic**: A bus over which nodes exchange messages in a publish-subscribe communication model within the Robot Operating System (ROS) framework. Nodes can publish messages to a topic or subscribe to receive messages from a topic. This allows for asynchronous communication between different parts of a robotic system. For example, a sensor node might publish data such as camera images or laser scans to a topic, and a processing node could subscribe to that topic to process the incoming data. Topics are unidirectional, meaning that data flows from publishers to subscribers, enabling modular and scalable communication in distributed robotic systems.
- **Mavlink**: A communication protocol used in drones and other unmanned vehicles to allow different components to communicate with each other. It works like a common language, enabling the exchange of information between the vehicle's flight controller and various software or hardware systems, such as ground control stations and onboard sensors. MAVLink ensures that commands, telemetry data, and status updates are transmitted reliably, facilitating effective control and monitoring of the vehicle.
- **rclpy**: A Python client library for ROS 2 that allows developers to write robotic applications in Python. It acts as a bridge between Python programs and the ROS 2 framework, enabling communication between different software components. Similar to how a Python library lets you use functions and classes in your programs, rclpy provides the tools needed to create nodes, send and receive messages, and interact with the ROS 2 ecosystem using Python. This makes it easier for developers to implement robotic functionality in a high-level programming language.
- **Python Runtime**: The environment where Python code is executed. It includes the Python interpreter, which reads and runs Python scripts, and manages the execution of Python programs by translating code into machine-readable instructions. This runtime environment handles memory management, error handling, and communication with the operating system, enabling Python applications to run smoothly across different platforms. It provides the necessary tools and libraries for executing Python code efficiently.

Table 1: List of Figures

Acknowledgements

I would like to express my deepest gratitude to everyone who supported me throughout this project. Special thanks to Ryo Koblitz, Matthew Andrews and Tugce Kobal from Nokia Bell Labs for guiding me through this project. Additionally, thank you to my supervisor Toktam Mahmoodi and to Yansha Deng for overseeing my work. I would also like to thank Luke Robinson for his help with the data generation and Eva Rouchut for proofreading my work. Their contributions were instrumental in the successful completion of this work.

2 Introduction

The rapid advancement of autonomous systems and robotics has ushered in new opportunities across various industrial domains, particularly in warehouse management. The integration of drone-based systems into these environments offers a promising future for enhancing operational efficiency, safety, and reliability. This report delves into the development and application of a Sim-to-Real framework designed to facilitate the deployment of autonomous drones for safety audits in warehouse settings. By leveraging cutting-edge simulation technologies, such as Nvidia's Omniverse and Isaac Sim, this project aims to bridge the gap between virtual models and real-world implementations, thereby providing a robust foundation for future research and practical applications.

This project is centered around two core objectives: the creation of a highly realistic virtual warehouse environment and the deployment of an autonomous drone-based safety audit system within this environment. The virtual environment serves as a critical tool for developing and testing drone strategies, enabling researchers to explore various scenarios and refine their approaches in a controlled, cost-effective manner. The subsequent deployment of these strategies in a real-world context demonstrates the viability of the Sim-to-Real approach, highlighting its potential to revolutionise warehouse operations.

The report is structured as follows: an exploration of the current limitations in existing solutions and the advantages of drone-based systems, a comprehensive review of relevant literature, a detailed methodology outlining the tools and techniques employed, and an evaluation of the results obtained from both simulated and real-world testing. The report concludes with an analysis of the legal, social, ethical, and professional considerations associated with the deployment of autonomous drone systems in industrial environments, emphasising the importance of responsible innovation in this rapidly evolving field.

3 Background

3.1 Aim of the project

This project aims to create a foundation for drone-based Sim-To-Real strategies in warehouses. This comprises of 2 main parts, the first one being the development and integration of a state-of-the-art suite of tools to tackle every important aspect of a simulation.

This includes:

- Creating a virtual warehouse environment as close as possible to the real world
- Import a Quadcopter in the virtual environment
- Build bridges to allow for external pieces of software to communicate with drones inside the simulation software for maximum compatibility with existing strategies
- Develop a customisable drone control system to interact with the drone from outside the simulation

All these tools are designed to allow researchers to develop their strategies as fast as possible by giving them a foundation to build on top of while keeping the simulation as close to reality as possible.

The 2nd and final part of this project is to utilise the created framework in a scenario to demonstrate the capabilities of the strategy. The chosen scenario is "Autonomous drone-based safety audit for warehouses", this will utilise all the previously mentioned tools to achieve a fully functional real use case for the framework.

3.2 Limitations of existing solutions

Current research focus on very specific aspects like Risk management [1], Reinforcement Learning [2] or Path Planning models [3] for example, however, there does not seem to be a solution that integrates state of the art tool for a true all in one simulation of drones inside warehouses. The absence

of an integrated framework stems from the inherently complicated nature of warehouses, where managing people, machinery, and robots in a fast-paced environment demands precise coordination to maximise efficiency. Therefore, developing a drone-based simulation software that considers all of these variables is challenging.

Another limitation regarding current solutions for robot warehouse management is the fact that they are usually based on cheap fixed cameras or expensive ground robots which heavily limits the visual coverage since cameras have dead angles and most of the ground robots cannot climb stairs. Even the most advanced "robot dogs" [4], heavily used for surveillance in recent years, cannot cover the entire volume of environments as tall as warehouses. A drone-based solution solves most of these issues while being far cheaper than the two options listed previously.

3.3 Advantages of Drones

In environments as dynamic and chaotic as warehouses, accidents and damages are regular occurrences, and ground robots are especially prone to being hit by forklifts or falling objects. The operating space of drones, at least 3 meters in the air, is usually completely different to everything else in a warehouse (humans, forklifts, ladders etc...), therefore, they are mostly safe from any obstruction or physical contact with most things operating inside a warehouse.

Drones can also cover a lot more space than ground robots, they are not limited to exploring along the X and Y axis of the floor, they can also explore the entirety of the Z axis as well, meaning they can do tasks regarding the top of shelves, ventilation pipes usually close to the roof, as well as regular ground surveillance and monitoring.

The lack of offers for a complete aerial robotics simulation framework could have also influenced the lack of drones implemented in warehouses, which this project aims to solve.

3.4 Expected Benefits

The implementation of this Sim-to-Real framework for autonomous drone-based is expected to yield numerous benefits:

- **More research**: The ease of implementation brought by this project could spawn an interest in researchers to develop fully integrated solutions based don't the framework.
- **More implementation**: An increase in academic research is usually the first step towards wider adoption. Additionally, the ability to replicate a real warehouse and to see a robust solution running inside a realistic simulation environment using this technique will instil confidence in the warehouse's managers to deploy aerial robotics for monitoring systems inside their facilities.
- **Safer Implementations**: Being able to more extensively test the techniques and validate them over thousands of simulation runs, will make for a much more robust approach and overall a safer implementation for people working with quadcopters flying above them.

Overall the benefits of this project are likely to significantly improve the domain of aerial robotics for indoor use in the logistics industry by making it more convenient, more efficient, safer and cheaper to develop fully integrated solutions.

3.5 Literature Review

This project is multi-dimensional, however, we can identify 3 main areas of research; Artificial intelligence-based drone control, Computer Vision for surveillance and Aerial Path Finding.

- **OmniDrones, An Efficient and Flexible Platform for Reinforcement Learning in Drone Control**[5]: This study introduces OmniDrones, a platform designed for reinforcement learning in drone control. OmniDrones is characterised by its efficient and flexible design,

facilitating the simulation of various drone application scenarios. It leverages GPU parallelism for high performance and offers a suite of benchmark tasks that serve as examples for further customisation. This platform supports a range of drone models, sensor configurations, and control modes, making it an ideal testing ground for RL algorithms aimed at enhancing drone capabilities in complex environments.

- **Sim2Real in Robotics and Automation: Applications and Challenges**[6]: A resource on MIT's DSpace that discusses Sim2Real strategies in robotics and automation, focusing on the challenges and applications of transferring models and control programs acquired in simulation directly to the real world. This concept, known as Sim2Real transfer, is crucial for leveraging simulation's benefits—such as safety, cost-effectiveness, and speed—while ensuring successful real-world application.

- **Sim2Real: Closing the Reality Gap in Robotics Simulation**[7]: This paper explores various techniques to close the "reality gap" in robotics, where simulation models and environments differ from real-world scenarios. It discusses how accurate modeling of physics, environment conditions, and sensor inputs in simulations can improve the reliability of transferring drone control strategies to real-world warehouse environments.

- **A Path Planning Model with a Genetic Algorithm for Stock Inventory Using a Swarm of Drones**[3]: Focuses on developing a mathematical model for conducting inventory tasks in a cluttered warehouse environment. The authors propose a method that controls drones capable of moving in all directions for imaging and data transmission. The solution enables automatic operation without external orientation devices, relying on pre-planned routes and drone movement. This research directly tackles the issue of pathfinding in a warehouse environment with the ability to operate without GPS or external sources of localisation. However, this approach seems to be made for static environments only since the routes are pre-planned, therefore, it cannot be the only basis for our strategy since we are targeting dynamic warehouse environments. Conclusions in relation to our project can still be drawn especially regarding the challenges of close-flight in warehouses.

- **Drones in Warehouses: Strategies for Autonomous Inventory Management**[8]: This paper investigates the use of drones in warehouse environments for tasks such as inventory management, mapping, and monitoring. It discusses the challenges and solutions for autonomous navigation in cluttered and dynamic environments, providing valuable insights for developing robust and reliable drone control systems.

- **An Adaptive Path Replanning Method for Coordinated Operations of Drones in Dynamic Urban Environments**[9]: Introduces an adaptive path replanning method that addresses dynamic changes in urban environments for drone operations to safely adjust their flight in real-time. This approach includes a discrete rapidly exploring random tree algorithm for path generation and demonstrates improved efficiency in handling large-scale problems with numerous dynamic changes. This method heavily relates to the problem at hand by allowing for fast path changes based on unpredictable events.

- **Risk Assessment for the Use of Drones in Warehouse Operations in the First Phase of Introducing the Service to the Market**[1]: Offers a new approach to risk assessment for drones in warehouses. The methodology, developed from a literature review, historical data, and expert knowledge, identifies ten scenarios of adverse events and analyses risk mitigation. While not being a technical state-of-the-art improvement, this research tackles something that is often overlooked in research: the safety impact of technical progress. Not taking this into account is one of the reasons great technological progress is not implemented in real applications; therefore, keeping this factor in mind from the very beginning of this project's journey is essential.

- **skrl: Modular and Flexible Library for Reinforcement Learning**[10]: An open-source Python library for reinforcement learning in various environments, including OpenAI Gym,

DeepMind, NVIDIA Isaac Gym, and Omniverse Isaac Gym. It speeds up the training by having multiple duplicating robots and making them learn in parallel. This library will heavily speed up the process of reinforcement learning in Nvidia Omniverse and could allow for complex and organic reactions to unpredictable events through the simulation of real-world situations, for example; workers roaming the facilities, forklifts driving, or ladders being carried around.

- **i-Sim2Real: Reinforcement Learning of Robotic Policies in Tight Human-Robot Interaction Loops** [2]: This paper discusses the Sim2Real transfer process in environments involving human-robot interactions, which are particularly challenging due to the unpredictability of human behavior. The study focuses on improving the robustness of policies developed in simulation before applying them in real-world scenarios, which is crucial for tasks like navigating drones in dynamic warehouse environments where human presence is common.

- **Evaluation of Techniques for Sim2Real Reinforcement Learning** [11]: This paper evaluates various techniques to bridge the "reality gap" in Sim2Real learning, which is the difference in performance when a model trained in simulation is deployed in the real world. The study's insights into handling discrepancies between simulated and real environments can be particularly beneficial for enhancing drone operations in warehouses, where environmental conditions can vary significantly.

- **Sim2Real Transfer in Deep Reinforcement Learning for Robotics: A Survey** [12]: This comprehensive survey covers various methods and challenges in transferring reinforcement learning models from simulation to real-world applications. It highlights the importance of domain randomisation and robust policy learning, which are essential for ensuring that drones can effectively operate in the complex and dynamic settings of a warehouse.

- **Accurate Dynamics Models for Agile Drone Flight: Zero-Shot Sim2Real** [13]: This research focuses on developing accurate dynamic models for drones to facilitate zero-shot Sim2Real transfer, where no additional training is required in the real environment. This approach is highly relevant for ensuring that drones can perform agile maneuvers in cluttered warehouse spaces right after being trained in simulation.

Papers \ Relevant subjects tackled	Sim-to-Real Strategies	Computer simulation	Indoor drones usage	Autonomous surveillance/monitoring	Computer vision	Path Planning	Reinforcement learning
Smart Drone Surveillance System Based on AI for Intrusion and Fire Accident Detection [14]			X	X	X		
OmniDrones, An Efficient and Flexible Platform for Reinforcement Learning in Drone Control [5]		X					X
Sim2Real in Robotics and Automation: Applications and Challenges [6]	X	X					
Sim2Real: Closing the Reality Gap in Robotics Simulation [7]	X	X					
A Path Planning Model with a Genetic Algorithm for Stock Inventory Using a Swarm of Drones [3]			X	X	X	X	
Drones in Warehouses: Strategies for Autonomous Inventory Management [8]	X		X				
An Adaptive Path Replanning Method for Coordinated Operations of Drone in Dynamic Urban Environments [9]					X	X	
Risk Assessment for the Use of Drones in Warehouse Operations in the First Phase of Introducing the Service to the Market [1]			X				
skrl: Modular and Flexible Library for Reinforcement Learning [10]							X
i-Sim2Real: Reinforcement Learning of Robotic Policies in Tight Human-Robot Interaction Loops [2]	X	X					X
Evaluation of Techniques for Sim2Real Reinforcement Learning [11]	X	X					X
Sim2Real Transfer in Deep Reinforcement Learning for Robotics: A Survey [12]	X	X					X
Accurate Dynamics Models for Agile Drone Flight: Zero-Shot Sim2Real [13]	X	X				X	

4 Methodology

This section is going to outline the tools and techniques used to achieve the desired goal outlined in the previous section.

4.1 Choice of tools

An in-depth analysis of the tools used in this project will be conducted and reasons for the choice of these specific tools will be given.

4.1.1 Simulation Environment

With the rise of AI-powered robotics in the past 10 years, the difficulty of developing such strategies solely in the real world has been more and more apparent as the goals for these AI-powered robots have become more complex. The reasons being that real-world robots are expensive and training artificial intelligence models by trial and error in the real world could lead to a lot of broken parts, expensive repairs and an overall slow process. Therefore, there has been an increasing demand for very realistic computer-powered simulation environments to develop and finalise these machine-learning models that could be then transferred into the real world without much modifications, this is the principle of Sim-to-Real (1). Lately, Nvidia Omniverse has been getting a lot of attention due to its very high visual fidelity compared to its competitors and its suite of specific environments with dedicated purposes, such as Isaac Sim for robotics simulations or Isaac Gym for reinforcement learning (1). For this particular application, the choice was made to use Isaac Sim, specifically for its extensive support of different robots. Isaac Sim presents itself more like a game engine made for robot simulation than a physics simulation software for robotics, which leads to a very high level of visual details compared to competitors such as Gazebo or Mujoco. Isaac Sim is the foundation for this entire project, it was used to build from the ground up the virtual warehouse environment and run all the simulations of the drones flying in the warehouse.

4.1.2 Types of Quadcopters

There are several options when it comes to importing a drone inside Isaac Sim:

- Build a drone from scratch in a 3D design software like Blender or Maya, set up the physics for the model (center of mass, weight, collision...) and import it in Isaac Sim
- Use a preexisting drone model, import it as an asset and build the control for the drone from scratch in python or ROS
- Utilise a pre-existing framework that links with Isaac Sim to import a drone and give you access to low-level control or pre-built control scripts

In the interest of compatibility, a pre-existing framework was used in this project: Pegasus Simulator. It provides a very realistic drone and allows users to connect to the drone using ROS, Mavlink or a custom communication protocol from outside of Isaac Sim. It will enable researchers using this project to use previously developed control software written for ROS(1) or Mavlink(1) with very little friction since it will run outside of Isaac Sim and act just like a real drone from the point of view of the control system.

4.1.3 Communication protocol

Pegasus Simulator offers ROS (1) and Mavlink (1) bridges to communicate with the drone inside the Simulation. Mavlink offers a more high-level control of the drone, meaning the user does not have to worry about the actual motor power inputs and balancing of the drone, it enables the user to send commands to change the angle, rotation or altitude of the drone and Mavlink takes care of translating these commands into motor power inputs. This allows for much faster and implementation of

control strategies. ROS allows for low-level interaction with the drone, meaning users can send direct instructions to each individual rotor to control rotation and altitude, which enables the development of a more customisable approach but significantly increases the implementation time.

While ROS and Mavlink are both industry standards, they are used in very different scenarios. To better choose which technique to use, a table comparing the strength of each protocol was made to provide a concise assessment of both techniques compared to each other.

Table 2: Comparison between Mavlink and ROS for Drone Control in Isaac Sim

Aspect	**Mavlink**	**ROS**
Ease of Use	High	Medium
Level of Control	High-level (abstracted)	Low-level (granular)
Support for GPS coordinates	Excellent	None
Customisation	Low to Medium	High
Development Time	Short	Long
Used in Reinforcement Learning	Yes	Yes
Integration with Existing Systems	Excellent (standardised)	Excellent (modular)
Realism of Control	Good	Very High
Support for Complex Tasks	Moderate	Excellent

The comparative table (2) clearly shows that both protocols have their differences and they can both be valid depending on the scenario. For example, if reinforcement learning is the goal, which usually requires low-level input to the drone's motors, ROS seems like the better option. However, if your technique involves controlling a drone using GPS positioning and expecting the drone to follow autonomously a sequence of action, Mavlink seems like the better protocol.

However, communication to give instructions to the drone is not the only type of communication needed in this project, there is also a need for transmitting the video stream from the point of view of the drone to the Artificial Intelligence strategy. ROS provides a suite of tools to analyse, pre-process and transmit video streams from cameras, which is not offered in such an extensive way through Mavlink.

Therefore, this project will use a combination of ROS and Mavlink, the former being responsible for steering the drone and the latter will be used to transmit and post-process the POV (point of view) camera feed from the front of the drone.

4.1.4 Collision Avoidance

In a cluttered environment, such as a warehouse, it is mandatory to avoid any collision with poles, boxes, shelves... Especially in the targeted scenario, it will fly above workers, which could lead to injuries in case of a crash.

The Pegasus Simulator drone can be equipped with a depth camera and a camera stream can be broadcast locally on the computer through a ROS topic(1). This video stream is then captured by a Python script using rclpy (1) which analyses the video frame-by-frame to detect any close objects and orders the drone to rotate away from the potential point of contact.

The depth camera stream from the ROS topic(1) is interpreted as a video where the scale of colours determines the distance (Dark Blue = close and Red = far away). A simplified version of the collision avoidance cycle is shown in figure 1:

- Step 1: An object (a pillar) has been detected, on the ROS depth video stream, within the distance threshold chosen by the user (1 meter in this case)
- Step 2: The Python navigation system determines if the object is to the left or to the right of the drone
- Step 3: The navigation system sends a message to Mavlink with a direction to steer the drone away from the point of collision

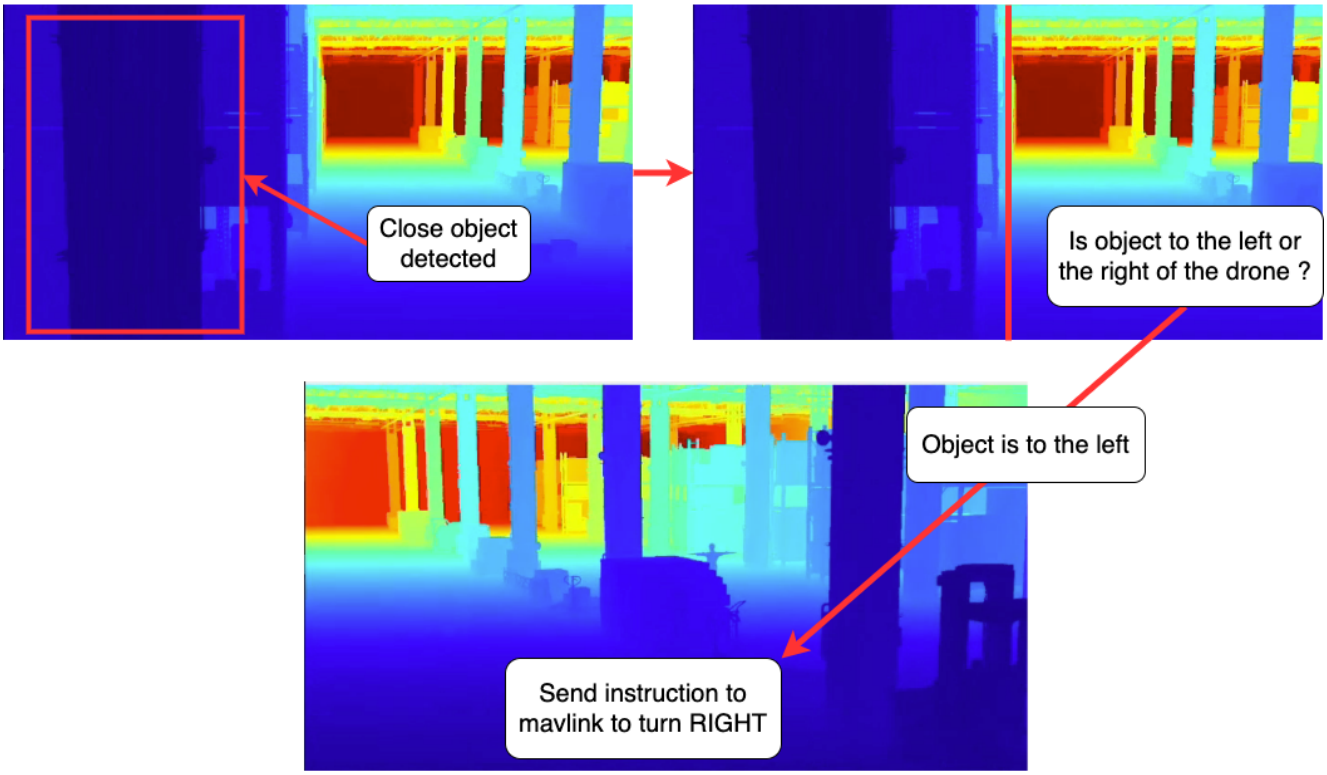


Figure 1: Collision Avoidance Detection Cycle

4.1.5 Navigation Protocol

A drone cannot go in a straight line from point A to point B in a cluttered environment. In these types of situations, two options are usually considered: developing a complex autonomous path-finding algorithm or having very clear instructions given by a human operator on where the drone needs to position itself.

The drone control strategy developed as part of this project is a hybrid solution: a waypoint system combined with automatic collision avoidance. Mavlink allows for GPS waypoint missions, meaning that the user defines a sequence of GPS locations that the drone will have to go through and in this case, the GPS locations will be translated to positions inside the virtual warehouse which is combined with autonomous collision avoidance for a semi-autonomous navigation system overall. In the case of a waypoint mission crossing through obstacles, there is, by design, a negative feedback loop between obstacle avoidance and a waypoint mission, where the navigation system will try to steer the drone towards the point of contact, while the obstacle avoidance will steer the drone away from it. This negative relationship is managed by a PID controller and more weight can be given to either the anti-collision or the Mavlink waypoint mission system and the resulting performance from that system will be examined in the result section.

The reasoning behind choosing a relatively simplistic system is the fact that researchers would easily be able to set their own mission for the drone without being experts in other types of navigation (SLAM, Beacon-Based Navigation, Lidar...) while still allowing for complex series of actions.

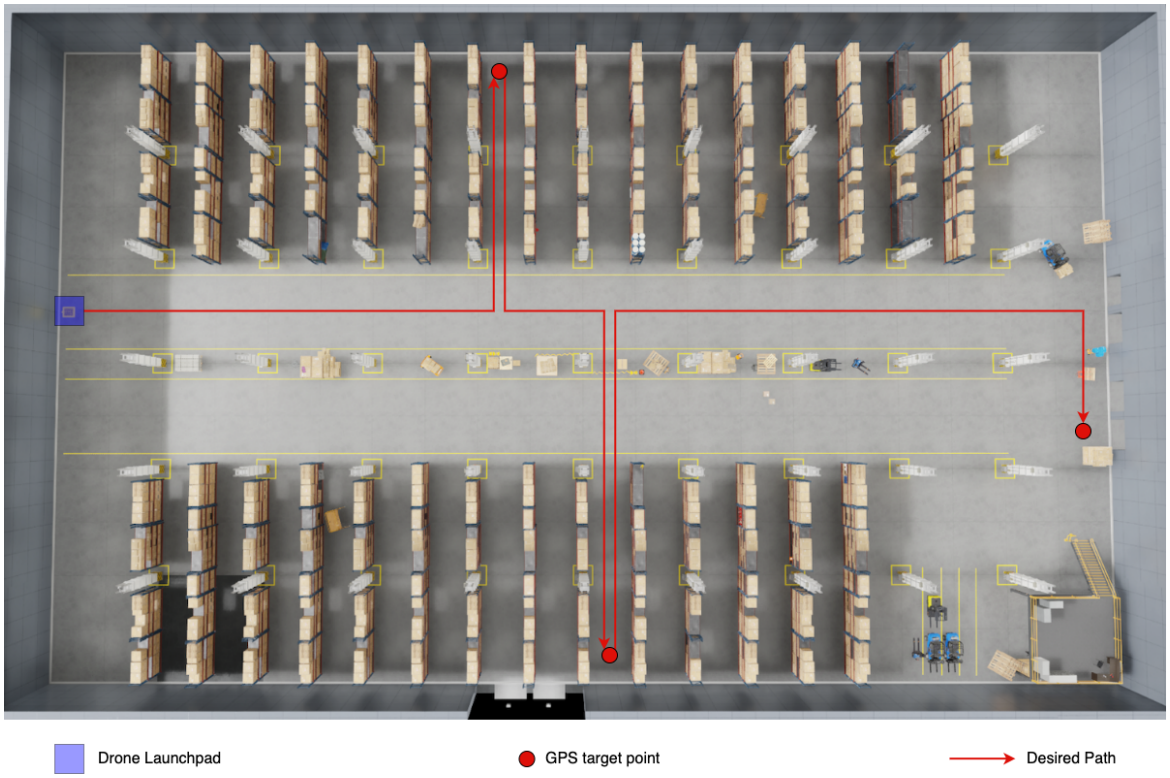


Figure 2: Example of a drone waypoint inside the warehouse

4.1.6 Computer Vision

The end goal of this project is to provide the ability for researchers to automatise tasks within a warehouse and full autonomy usually involves Artificial Intelligence. Therefore, a comprehensive computer vision strategy is a mandatory part of this project and in order to provide such option as part of this framework there are two main workloads that researchers absolutely need to develop Computer Vision strategies:

- Dataset Creation: involves taking thousands of pictures of a specific object or scenario and label each image accordingly to its features. For example, suppose the strategy is to go around a warehouse to detect unsafe storage (open containers, parts of boxes hanging of shelves...). In that case, researchers will have to create a dataset containing thousands of pictures of these scenarios and label some as safe and some other as unsafe. This is usually a long and tedious process that this project will aim to minimise by providing tools to automate the dataset generation.

- Model: the choice of models is extremely important and can make massive differences depending on the scenario. Therefore, to allow for researchers to have as much freedom as possible, the aim is to provide a flexible strategy to allow for a wide range of models to be compatible with both the dataset and the video stream from the simulation.

To help with the creation of datasets, Isaac Sim provides a library called Replicator, which allows for the creation of an additional camera in the virtual environment and automatic labelling of objects. The position and orientation of this camera can be controlled by a Python script which enables users to code the movement of the camera, take pictures of a specific object and label desired objects in the picture. It also allows for the Python script to change the state of objects in the scene while generating the dataset, for example, while moving the camera at different angles Python can make a box change colour, texture or lighting effects which has the added benefit of a more varied dataset and better generalisation from a model when trained on the dataset. If programmed correctly, Replicator empowers users to create a dataset comprising thousands of custom-labelled pictures of an object which can be used to train a Computer Vision model.

The choice of the computer vision model is extremely important and will be different for each research application will use different models and AI frameworks. Therefore, 2 codebases will be provided as part of this framework to use Tensorflow or Pytorch (the 2 main AI frameworks) with examples of how to use them with a dataset generated with Replicator.

4.2 Creation of the Virtual Environment

Following the deep assessment of the tools and their strength in the previous section, here the focus will be on detailing how each tool was used to create a simulation environment inside Isaac Sim.

To accelerate the development of solutions inside their simulation software, Nvidia offers a wide range of simulation environments such as hospitals, data centres, factories, company offices and warehouses. However, those environments are very useful to start getting familiar with Isaac Sim and basic simulations but struggle to encompass the variety and complexity of real environments, therefore, as part of this project, an entire warehouse was built from the ground up using free and open licence 3D assets.

4.2.1 Warehouse shell

Isaac Sim, like many other simulation software, starts with an empty 3D environment with a plane along the X and Y axis where models can be placed. Alongside full environments, Nvidia also provides a very extensive library containing its own assets as well as assets from its partners. This library contains around 1700 3D models related to the warehouse which includes the starting point for this project: the floor, roof and walls of the warehouse.

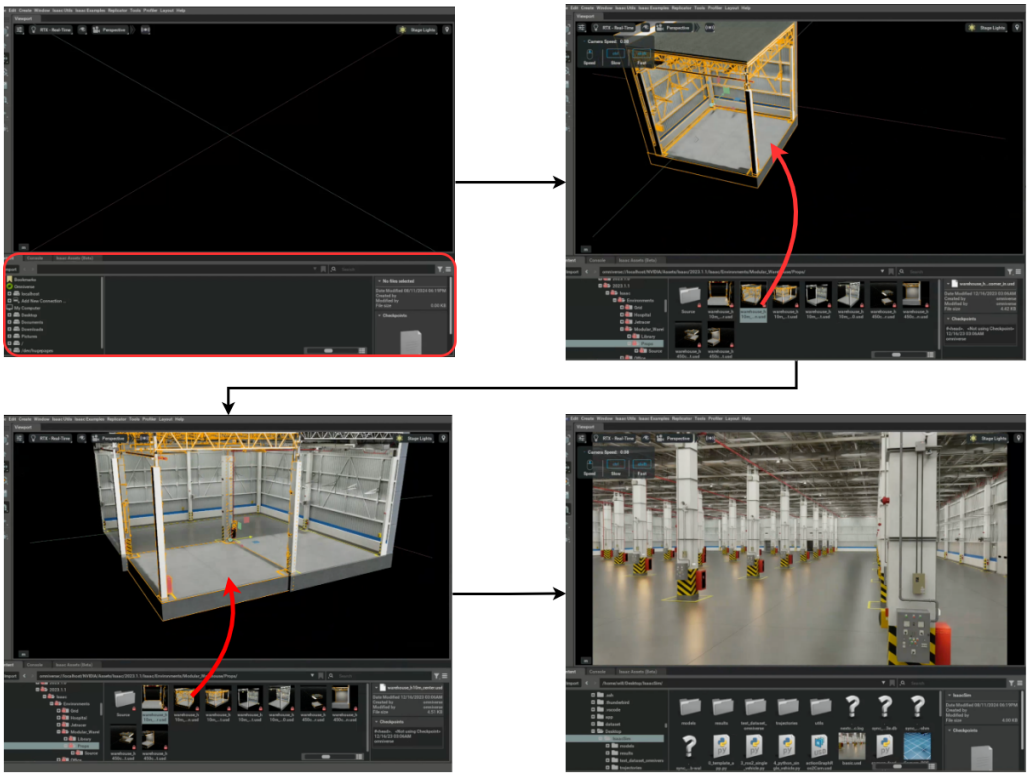


Figure 3: Process of Creating the Warehouse in Isaac Sim

The first step in figure 3 represents the empty Isaac sim environment in the top three-quarters of the screen, the bottom last quarter is the window where the different assets from the asset library can be accessed. The second step illustrates how each warehouse tile was placed inside the simulation, after choosing the desired asset, it is dragged and dropped inside the above window to place it on the 3D plane, each tile has to be placed, positioned and oriented by hand to match the placement and orientation of each tile already present in the simulation. The final image in figure 3 shows

the final warehouse shell with all the individual hand-placed tiles containing characteristics of a real warehouse: roof, walls, pillars, electrical infrastructure, fire systems, ventilation tubes...

4.2.2 Populating the Warehouse

The creation of the warehouse shell is the basis for building a full warehouse capable of being used in Sim-to-Real strategies, real warehouses are extremely busy, usually uneven, sometimes poorly organised and frequently containing various items dispersed haphazardly. The hectic nature of warehouses is the reason for their difficult virtual reproductions, therefore, in this project, every single item inside the warehouse (shelves, boxes, tubes, cones, signs, ladders...) was carefully hand-placed, in a similar fashion to the creation of the warehouse shell they were each orientated and organised based on real images to capture as much of the chaotic nature of warehouses.

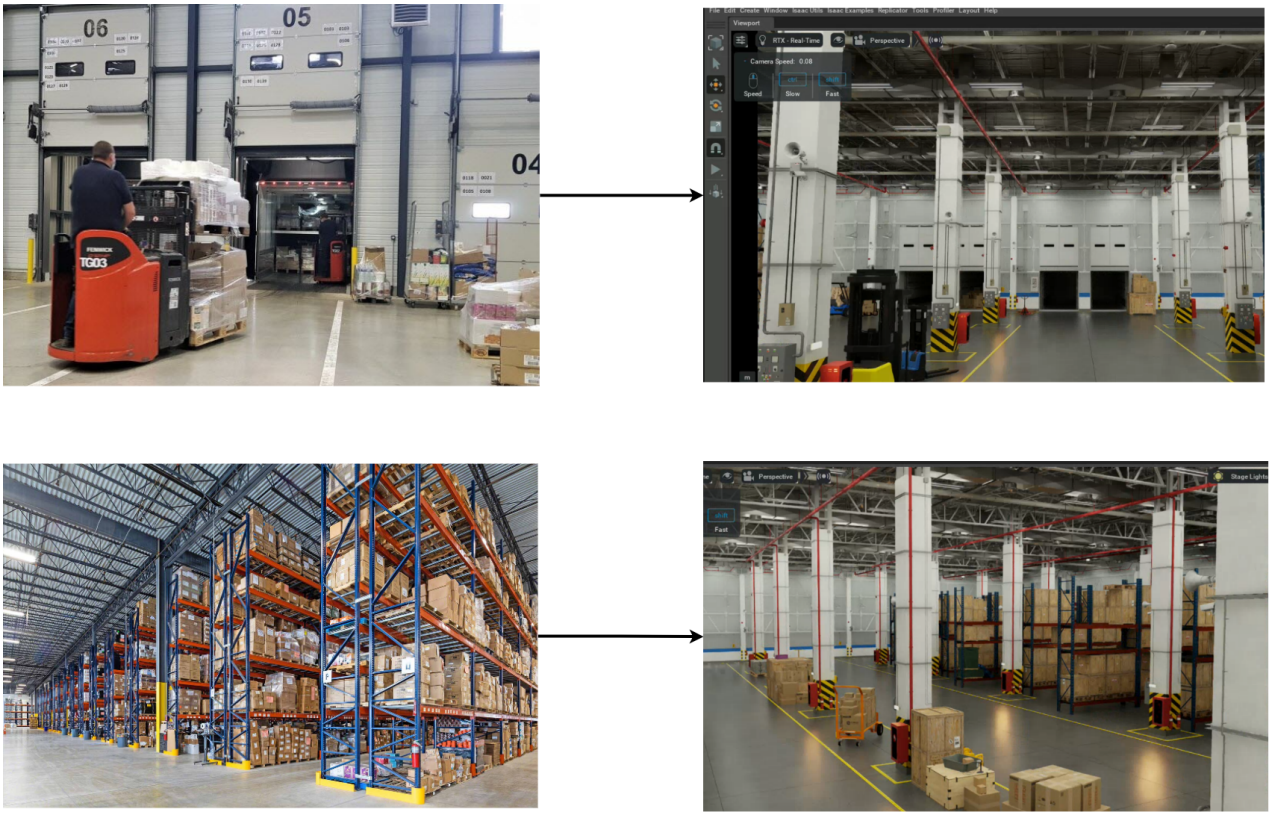


Figure 4: Comparison of Real warehouses and 3D Environment

Figure 5 showcases the similarity between real warehouses and the virtual one in Isaac Sim. The top 2 pictures clearly show how the truck loading area was repodused. In the real warehouse, pictured on the left, a wide range of elements can be seen, such as forklifts, special garage doors for trucks, boxes of different sizes scattered on the ground and safety strips on the floor. All these elements can be observed in the virtual reproduction with a relatively high level of fidelity. In the second set of images, similarities can also be observed with how various boxes (wooden, cardboard, wrapped...) are organised on shelves, additionally, empty shelves are also present on both with some containing randomly placed items.

Placing all these items by hand was primordial to mimic an environment where people are susceptible to leaving stuff around and not following exactly perfect procedures. This approach was critical in creating a realistic scenario that mirrors the natural tendencies of human behaviour, where perfect procedures are not always followed, and some level of disorder is very likely.

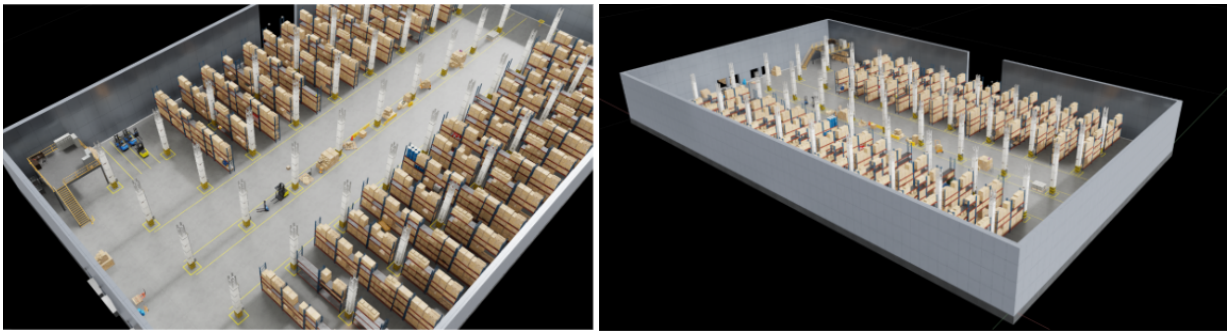


Figure 5: Final Layout of the Warehouse (roof off)

4.3 Implementation of the ROS Camera System

As discussed earlier in section 4.1.3, the communication protocol of choice for streaming the video of the drone's camera is ROS. Isaac Sim allows to pre-process the camera stream and makes it available outside of the simulation environment through the ROS bridge via a ROS topic, the stream for the topic can then be picked up by Python to do both the Computer Vision and the obstacle avoidance. In order to create such a system the creation of an Action Graph is necessary, it is a tool provided by Nvidia as part of Isaac Sim that allows to do visual scripting, meaning that instead of coding with text, blocks acting as functions are connected to create a visually flowing script.

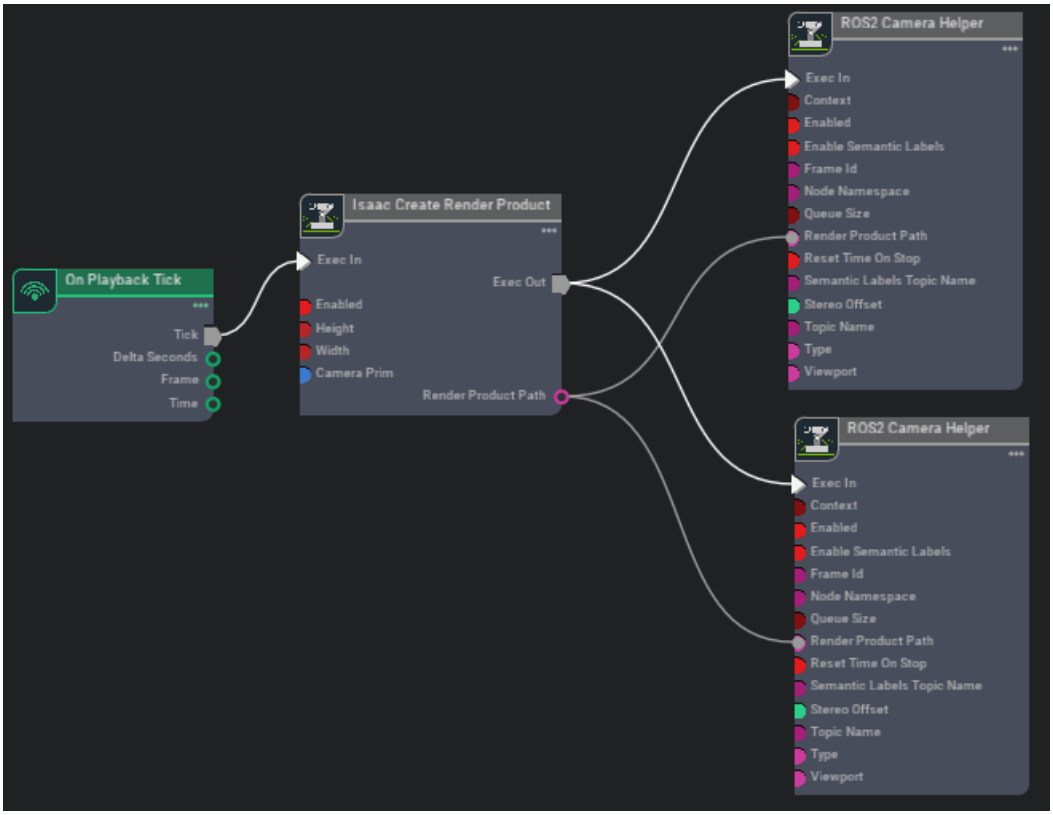


Figure 6: ROS Action Graph

Figure 6 shows the Action graph responsible for passing both the RGB camera and the depth camera for Computer Vision and collision avoidance respectively. The first node on the left, On Playback Tick, gives the execution rate (or "tempo") for the script, a playback tick in a simulation works similarly to seconds in the real world, at every simulation tick the entire physics and rendering are updated based on the state of the previous tick and the acceleration of objects. For example, for every tick, the position of the drone is updated based on its position in the previous tick and its

speed, the scene is then rendered again with the new information from the latest tick.

The second node, Isaac Create Render Product, is responsible for creating a camera instance inside the simulation, it needs to be attached to a 3D object inside the simulation, in this case, it is attached to the front of the drone and creates a video stream from the point of view of the drone. This video stream is refreshed every simulation tick through the playback node with the new position of the drone.

The final two nodes, both named ROS2 Camera Helper, get as inputs the simulation tick passed on from the previous node and a rendered product path. The latter refers to the POV (point of view) camera from the drone, they both post-process the video and pass it onto the rest of the computer through a ROS topic accessible through Python. The reason for having duplicated ROS2 Camera Helper functions is because they are both responsible for different post-processing on the drone's video feed. The first one creates a "depth" topic, which is a special camera stream with depth information: Blue/Dark colors representing close objects and Red representing far away objects. The second Action Graph node is responsible for passing on a regular video feed. Both can be seen, in figure 7, displayed through Python and Open-CV as Depth Camera and Video Stream windows respectively.

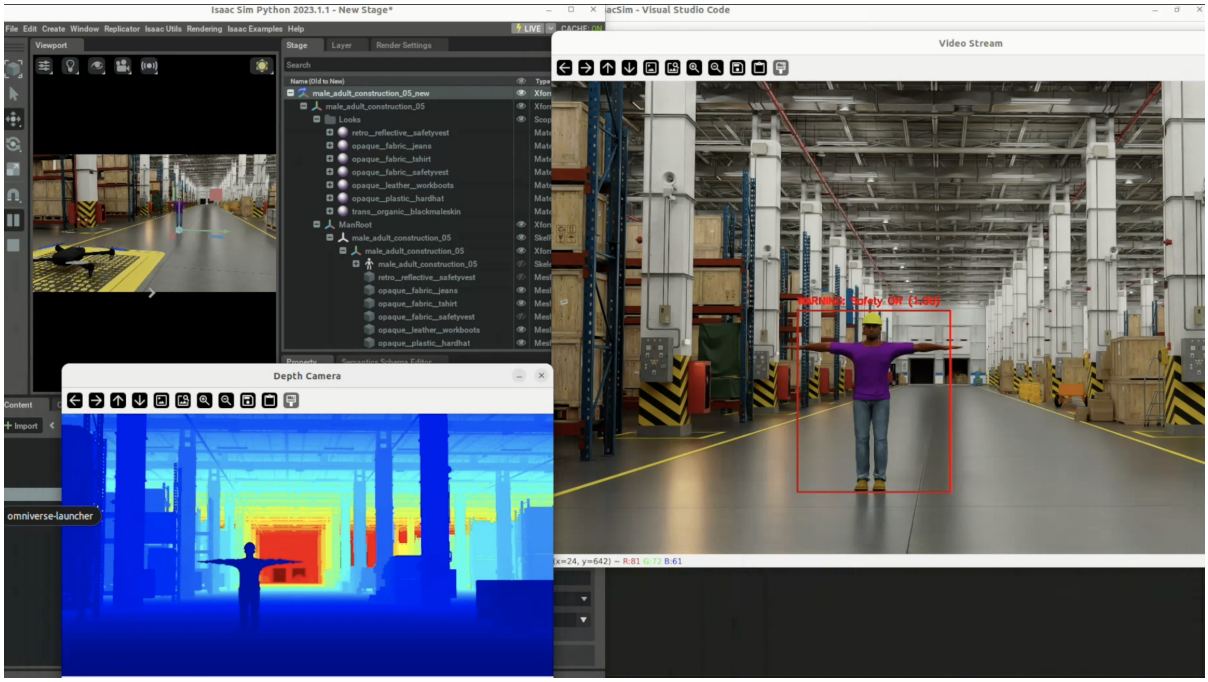


Figure 7: RGB video stream and Depth streams

4.4 Safety Audit Implementation

As stated in the section 3.1, providing an example use case for the project by utilising the developed tools is mandatory to show the usability, performance and benefits of the fully integrated software suites provided in this project. Additionally, utilising this framework as both a developer and a user will also help give a broader sense of its limitations and benefits. The use case that will developed here is "Safety gear detection in warehouses", which will involve an autonomous drone roaming the virtual warehouse and detecting workers without safety hats. This project will utilise the warehouse facility, the Mavlink (1) drone control system, the collision and avoidance system, the custom dataset generation to create a workers hats-on/hats-off dataset and finally the ability to make an off-the-shelf Computer Vision model, trained on the custom dataset, to interact with the simulation environment.

On figure 8, the entire structure is laid out. There are three main parts, the Isaac Sim virtual environment, the Python Runtime and the connectors that allow them to interact (Mavlink and ROS bridges). The modular approach of the framework allows for each part to be modified, and as long

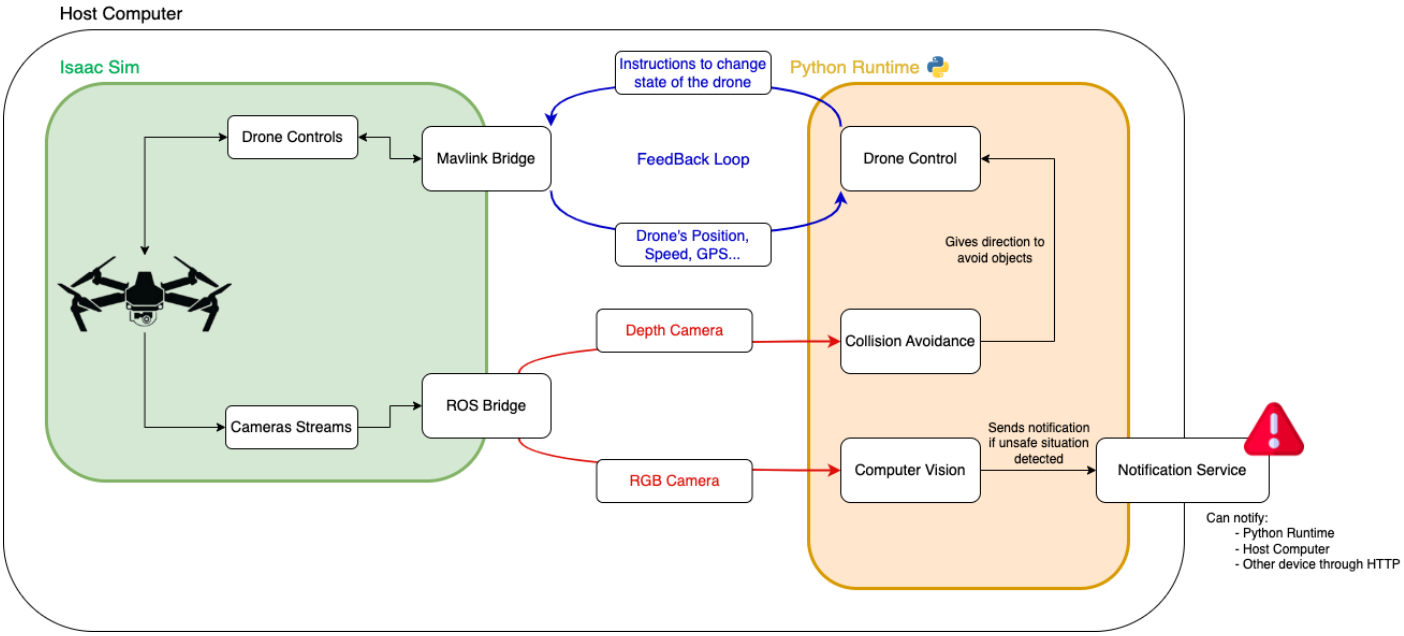


Figure 8: Network Diagram of Safety Audit Implementation

as the communication are not completely changed, the other parts of the project don't need to be modified. For example, since Mavlink and ROS are used in real world application, the drone inside Isaac Sim could be easily replaced by a real drone in a real warehouse and from the point of view of the Python Runtime there won't be any changes, the ROS and Mavlink bridges act as a layer of abstraction which allows for maximum flexibility. Being able to easily transfer from virtual to real world is key to achieve a true Sim-to-Real strategy.

4.5 Dataset Creation

Since the warehouse environment, drone and drone control strategy are already given with the framework, this section is going to focus on the Computer vision aspect of the project to determine if workers are wearing their safety gear. The first step towards this goal is to create a custom dataset using the Replicator example provided in the framework and adapt it.

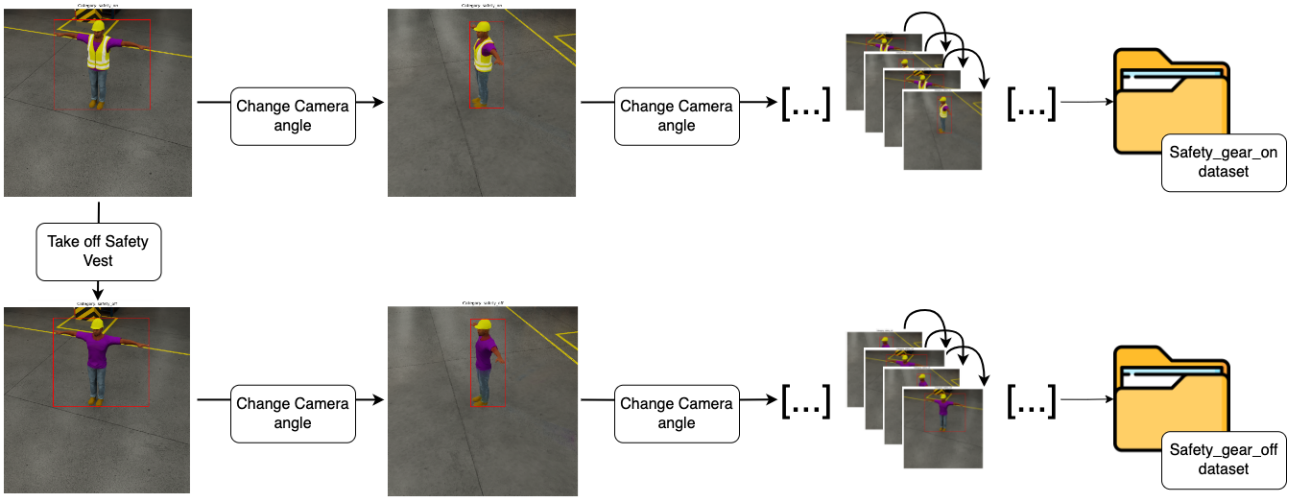


Figure 9: Dataset generation

In the Python Replicator Code (9), the different steps of the utilisation of the Replicator module can be identified:

- 1) Create a camera
- 2) Import a 3D model of a worker
- 3) Set up the Writer, responsible for saving and labelling images to the computer
- 4) Launch the code to trigger on each frame (for 10 000 frames) to change the angle of the camera
- Save each image and its bounding box into a dedicated folder that will serve as a dataset

In Figure 9, a visual workflow of two Replicator data generation code instances can be observed, one for a worker with a safety vest and one without a safety vest.

Those two datasets of images will allow for the training of a Computer Vision model that will be able to differentiate the workers with and without safety vests.

4.6 Computer Vision Model Selection and Implementation

In order to properly choose the most adequate model for this particular use case, the needs that the model has to fulfill have to be properly laid out first. In this scenario, the model needs to process a real-time camera video stream and detect workers without safety vests in the image. This task falls under the category of object detection and classification, which is fundamentally a pattern-matching problem.

4.6.1 Model Architecture

Given the nature of the problem, the decision was made to use a Convolutional Neural Network (CNN). Specifically, the ResNet50 architecture was selected for its proven performance [15] and computational efficiency. The ResNet (Residual Network) architecture, introduced by He et al. [15], is known for its ability to train very deep neural networks effectively by using skip connections, also known as shortcut connections.

The ResNet50 model consists of 50 layers, including:

- An initial convolutional layer with 64 filters of size 7x7
- A max pooling layer
- 4 blocks of convolutional layers, with each block containing multiple residual units
- A global average pooling layer
- A fully connected layer with softmax activation for classification

The residual units in ResNet50 allow the network to learn residual functions with reference to the layer inputs, rather than learning unreferenced functions. This approach addresses the degradation problem that occurs when very deep networks start to converge, enabling the training of much deeper networks than was previously possible.

4.6.2 Model Adaptation

To adapt the ResNet50 model for our specific use case of detecting workers without safety vests, we modified the final fully connected layer to output two classes: 'compliant' (worker with vest) and 'non-compliant' (worker without vest). The weights of the convolutional layers were initialised using pre-trained weights from ImageNet [16], a technique known as transfer learning, which typically results in faster convergence and better generalisation.

4.6.3 Data Preprocessing

As seen in Figure 8, the Python Runtime environment receives the drone's video feed from a ROS topic. Before passing the images to the ResNet50 model, several preprocessing steps are performed using OpenCV:

1. Resize the input image to 224x224 pixels, the standard input size for ResNet50
2. Normalise pixel values to the range [0, 1]
3. Perform data augmentation techniques such as random horizontal flips and slight rotations to improve model generalisation
4. Convert the image from BGR to RGB color space (OpenCV uses BGR by default, while most deep learning models expect RGB)

4.6.4 Model Training

The model was trained using a dataset of 16,000 synthetic images generated using the Replicator tool in Isaac Sim. The training process involved:

- Using a batch size of 24 images
- Training for 10 epochs
- Utilising the Adam optimiser with an initial learning rate of 0.001
- Using categorical cross-entropy as the loss function

4.6.5 Inference and Post-processing

During inference, the model processes each frame from the video stream. The output of the model is a probability distribution over the two classes ('compliant' and 'non-compliant'). A softmax function is applied to convert these raw scores into probabilities. If the probability of the 'non-compliant' class exceeds a threshold (set at 0.7 after experimentation), the system flags the frame as containing a worker without a safety vest.

To reduce false positives and account for potential momentary misclassifications, a temporal smoothing technique is applied. This involves maintaining a sliding window of the last 5 frames and only triggering an alert if the majority of these frames are classified as 'non-compliant'.

4.6.6 Performance Considerations

The ResNet50 model, while deep and powerful, is also computationally efficient. On our test hardware 9, the model achieves an inference time of approximately 15.82ms per frame, allowing for real-time processing of the video stream of more than 60 frames per second. This aligns with the project's requirements for a responsive system capable of detecting safety violations in real time.

Overall, the choice of ResNet50 aligns with the project's goal to create a robust, flexible, and efficient computer vision model that enhances the autonomy and functionality of drone-based systems in warehouse environments. Its combination of depth, efficiency, and adaptability makes it an ideal choice for the challenges posed by sim-to-real applications in dynamic and complex settings. The model's ability to process frames in real-time, coupled with its high accuracy (as will be discussed in the Results section), provides a solid foundation for the safety monitoring system.

5 Results

This section will focus on performance metrics regarding every part of the framework, considering performance inside the simulation, the transferability of the framework to the real world and the ethical implications of such framework with tangible metrics.

5.1 Optimisation performance on warehouse environment

Throughout the development of the digital warehouse environment, efforts were made to limit the computational cost of loading the environment and running simulations inside the warehouse. Considering the price of hardware to run such simulations, lowering the computational barrier of entry for this framework is mandatory to maximise the adoption and minimise the monetary cost for researchers wanting to utilise it.

In order to quantify the impact of the measures taken in that regard, 3 main metrics were tracked: FPS, Load Times and Memory Usage. An exhaustive list of the technical specifications of the computer used for these tests can be found here 9 and will give potential users an idea of what performance they could expect, based on their computer's specs, when running the framework inside Isaac Sim. Since performance will vary drastically depending on the hardware it is running on, it is important not to look at the actual hard numbers (FPS counters, load times in seconds...) but more at the improvements in percentage between the different scenarios.

5.1.1 Frames per second

FPS (Frames Per Second) refers to the rate at which the simulation renders and updates visual frames. It measures how smoothly and quickly the simulated environment is displayed and updated, affecting the fluidity of motion and responsiveness of the simulation. Higher FPS generally indicates better performance and a more realistic experience. it is primordial to make efforts towards optimising the smoothness of the simulation environment since low FPS can lead to unexpected behaviours within the simulation, For example: objects not being rendered fast enough causing collisions or Desynchronisation between the visuals and the data sent through Mavlink causing wrong decisions from the drone control software.

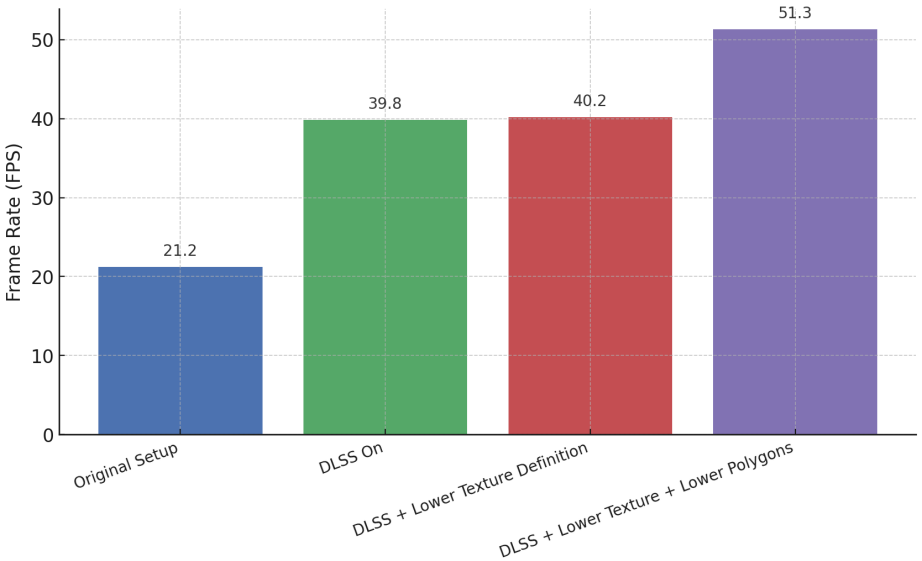


Figure 10: FPS Comparisons under Different Settings (higher is better)

Looking at the bar chart in figure 10, significant improvements can be seen from enabling DLSS (Deep Learning Super Sampling) within Isaac Omniverse. DLSS is a video upscaling technology provided by Nvidia; it operates by having the video card render the 3D scene at a reduced resolution to boost FPS. Subsequently, the video output is upscaled to enhance the clarity and sharpness of the image. This led to an increase in performance of 85.71%, however, it introduced some small visual artefacts like parts of textures glitching or undesired sparkling effects which need to be considered but it does not seem to impact the object recognition model in the implementation.

Lowering the texture quality had practically no effects on FPS, however, it made a substantial difference on the memory utilisation, which will be looked at in section 5.1.3.

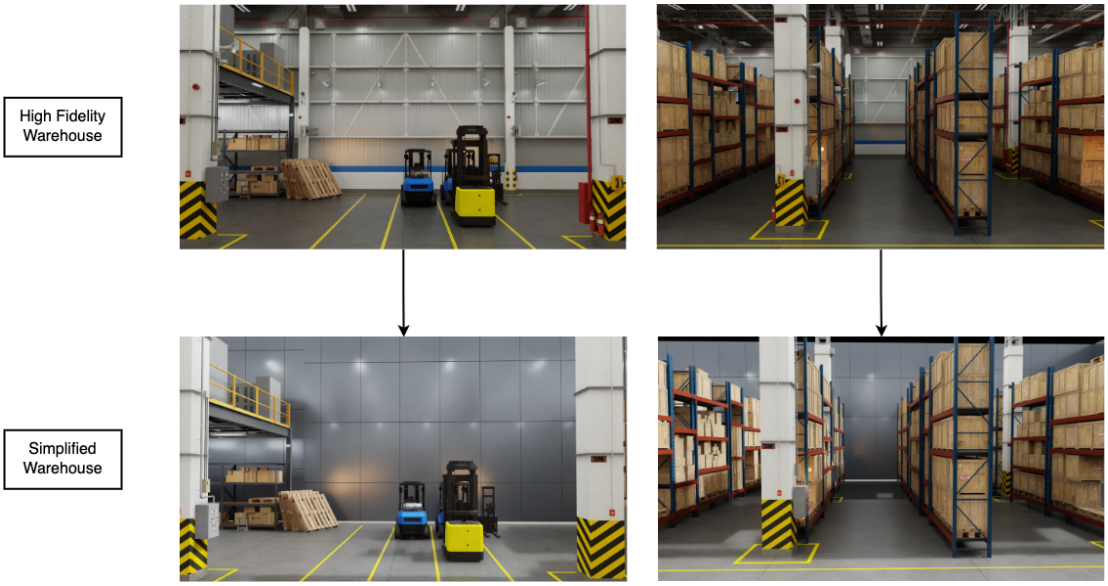


Figure 11: Realistic Warehouse compared to Lower Polygon warehouse

Finally, since FPS is solely based on how many 3D objects are to be rendered in the scene, providing an additional version of the warehouse environment with fewer details (Lower Polygones 11) helped increase the FPS by 28.25%. The details taken away from the warehouse environment are mainly details: Air conditioning pipes, fire pipes, electrical wires, roof details... This leads to much better performance while still retaining all the important objects (pallets, boxes, racks, safety cones...). A visual comparison of those two warehouses can be seen in figure REFERENCE!!!!!!

5.1.2 Loading Speed

Load times are the time in seconds required to start the simulation environment. This depends on CPU and Hard Drive performance, lowering the load times will make it easier for every computer to use this strategy, hence broadening its adoption.

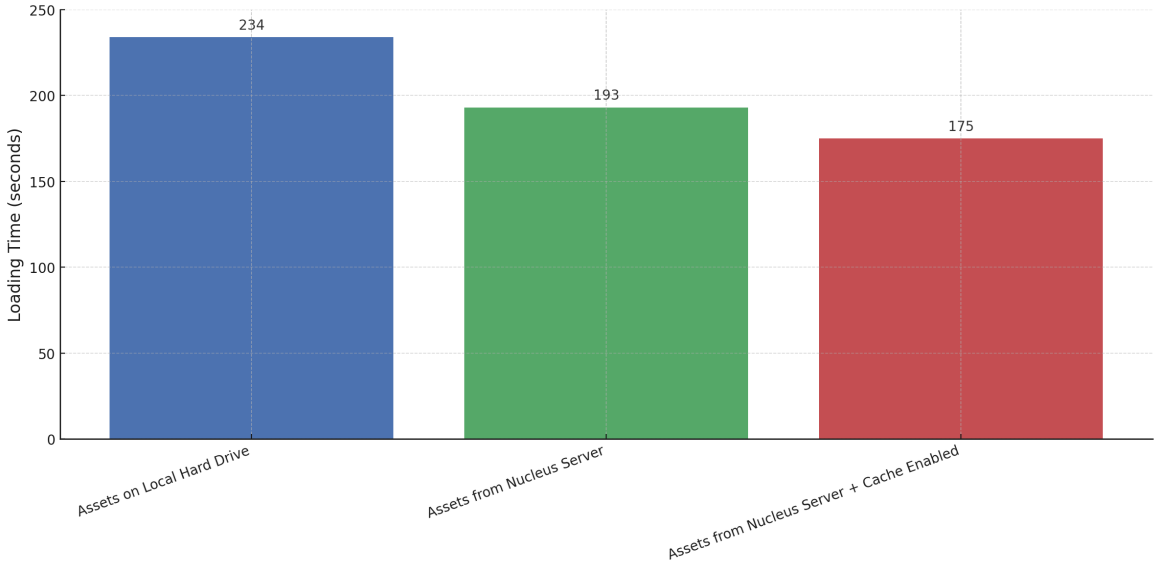


Figure 12: Impact of settings on loading times (lower is better)

Unlike FPS which is dependent solely on the GPU, optimising load times is a compounded problems. CPU, Hard Drive or even where the files are stored on the computer can significantly impact the wait times for the simulation to load. After many trials and errors, 2 main set-up

modifications made a meaningful impact. The first one is using a Nucleus Server, software provided by Nvidia to host files on a local server directly connected to the simulation environment, which sped up the process by 17.52% (12). In addition to using a Nucleus server, a different software provided by Nvidia as part of the Omniverse tools is Cache, responsible for storing in fast accessible memory the most often used assets, which led to an improvement of 9.33% (12) over just using Nucleus server.

Investigating the benefits of Cache further showed a substantial 69.71% (18) improvement in reloading times in the event of a simulation crash. Since crashes and bugs are very common in such large and complex simulation environments, being able to promptly reload the simulation environment is extremely important to speed up the development of research utilising this project.

5.1.3 Impact on Memory

The Memory Usage metric is the amount of memory space taken by the simulation on both the RAM and VRAM (CPU memory and GPU memory respectively). Computers with high amounts of memory are very expensive, therefore, lowering the required memory threshold follows the cost-effective mindset of this framework. Unlike FPS or load times, memory utilisation is the only metrics that would be the same across every computer since it is how much physical space the simulation software is taking, therefore, it is a very good indicator for the minimum specs required to run this framework.

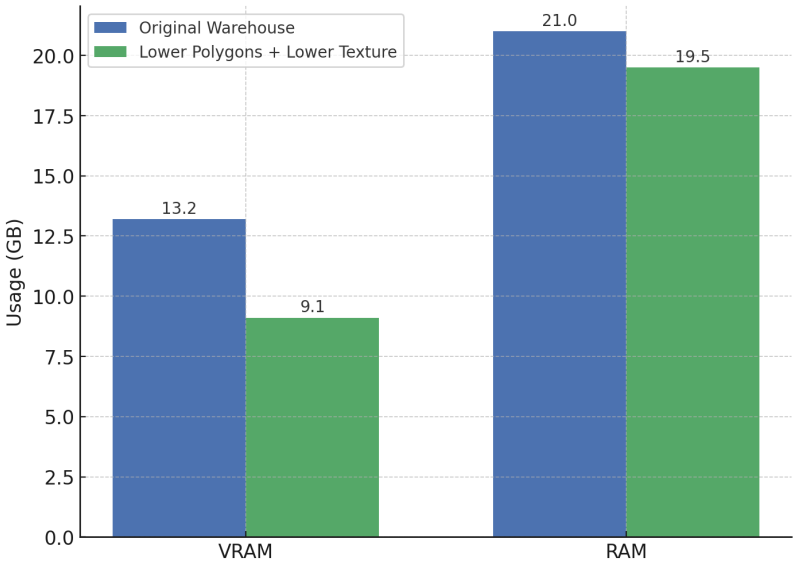


Figure 13: Memory Utilisation (lower is better)

Results present on figure 13 are the reported memory utilisation for GPU (VRAM) and CPU (RAM) during the full simulation with ROS topics, Mavlink and Computer vision running to be as representative as possible to performance in a real use case. Amongst all the explored optimisation techniques using the simplified warehouse (Lower Polygones 13) and lowering the textures were the only ones with tangible impact on the memory. The most noticeable improvement is on the VRAM (GPU memory) with a decrease of 4.1GB, which puts the memory requirement just under 10GB. Since these results are dependant on the simulation settings and not on the hardware, they can be utilised as hard numbers to spec out an appropriate machine to run the worker's safety vest detection experiment or other similar strategy. To quantify the impact of having the memory requirements just under 10GB, an understanding of the GPU market is required: most consumer-grade GPUs have either 4GB, 8GB 10GB or 12GB of VRAM, to get more than 12GB it is usually required to buy more expensive professional-grade GPUs, therefore, lowering the VRAM requirements from 13.2GB to under the 10GB threshold allows potential users of this framework to use less expensive and more readily available consumer-grade GPUs. This makes a major difference in favour of wider adoption and lower cost of entry, which is very beneficial for the autonomous drone-based research community.

5.2 Drone Control

Thoroughly evaluating the performance of the drone control system is primordial to give a comprehensive overview of this project's capabilities.

5.2.1 Waypoint Mission Accuracy

This section will focus on the performance of the navigation system through the analysis of the drone's performance in one main parameter: the balance between Collision Avoidance and Hitting the target point. Focusing on mission accuracy, crash rates, and lost drone incidences, the study explores how different weights on the navigation and collision avoidance systems affect these outcomes.

The technical definition of all 3 metrics is as follows:

- Crash: is determined by the drone position being recoverable (upside down, stuck in a corner) or by a short and rapid acceleration which would signify bouncing of an object (onboard accelerometer data through Mavlink)
- Hitting Target: is defined as being within 5% above or below (10%window) in X Y coordinates
- Lost Drone: signifies that after a defined amount of time (5x the average time to hit a target) the drone is considered lost, which, from observations, usually means stuck in an endless loop of avoiding all the obstacles around it

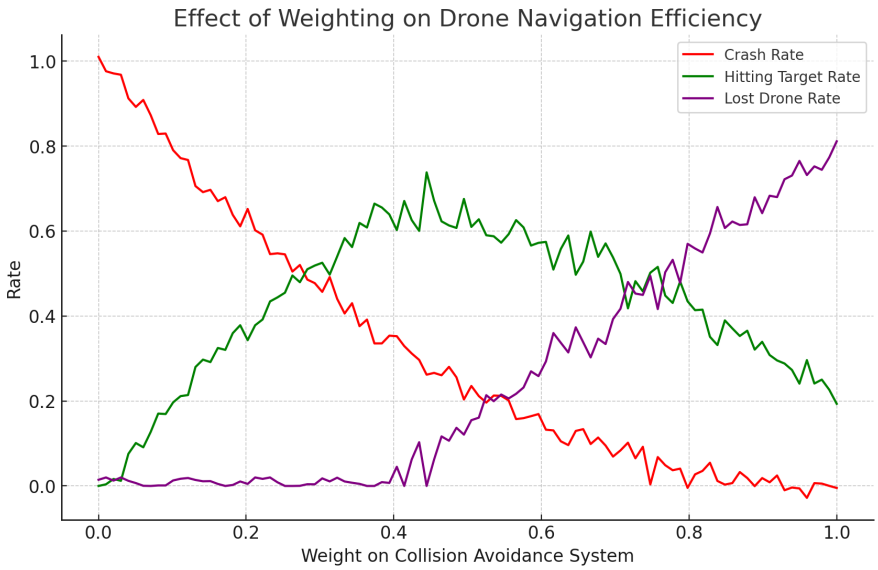


Figure 14: Impact of PID weight balance on drone control performance

Figure 14 illustrates the relationship between the collision avoidance/waypoint balance and three critical metrics: crash rate, hitting target rate, and lost drone rate. The graph is essential for understanding how effectively the drone manages its mission objectives depending on the weight balance between the waypoint mission navigation system and the collision avoidance system. The abscissa only represents the weight of the collision avoidance system, however, it is a binary system, therefore, the weight of the waypoint system is equal to 1 minus the collision rate weight.

To better understand this complex relationships between the 2 weights and the 3 metrics, a detailed breakdown of each metric and how they evolve with the changes in weight balance is necessary:

- Crash Rate: The crash rate decreases sharply as collision avoidance weight increases. This rapid decline suggests that enhancing collision avoidance systems significantly reduces crash probabilities. This behavior is expected as the collision avoidance system becomes more prioritised, enabling the drone to better evade obstacles.

- Hitting Target Rate: The hitting target rate shows a steady increase, in the beginning reflecting an improvement in mission success as collision avoidance systems start to counteract the navigation system that directs the drone in a straight line from A to B. However, it slowly decreases as collision avoidance takes more and more priority, meaning, the drone no longer takes any risks to reach its target which leads to incomplete missions.

- Lost Drone Rate: Initially, the lost drone rate remains low for weights ranging from 0.0 to 0.4. Beyond this point, the rate begins to increase, illustrating a gradual trade-off as the collision avoidance system continues to dominate, leading took reduced risk-taking, at the expense of precise navigation.

The data indicates that while an emphasis on collision avoidance drastically reduces crash rates, it is crucial to balance the weights to avoid compromising hitting the target. The lost drone rate's initial stability suggests effective navigation at low to moderate collision avoidance settings. However, as the collision avoidance system becomes too dominant, the potential for mission deviations increases, highlighting the importance of a balanced system that prioritises both collision avoidance and accurate navigation. The balance of this particular system seems to be for weights ranging from 0.4 and 0.5 for the collision rate avoidance (meaning 0.6 to 0.5 for the waypoint navigation) to maximise the rate of successful missions at around 70%

To conclude, this analysis demonstrates the importance of fine-tuning system weights to optimise drone performance, achieving a harmonious balance between safety and mission accuracy

5.3 Computer Vision

This section evaluates the performance of the Computer Vision model during training, inside the simulation as well as in real-world performances. The focus is to assess the model's ability to accurately detect and differentiate targets within the operational parameters set by the project while giving thoughts for improvements.

5.3.1 Performance During Training

During the training phase, the Computer Vision model was subjected to a diverse dataset created using the Replicator tool within Isaac Sim. The training process aimed to allow the model's accuracy and generalisation capabilities in detecting workers' safety vests in a warehouse environment.

- **Dataset Composition**: The dataset comprised of 16 000 labelled images with a 50% split between workers with and without a vest captured from different angles and under varying lighting conditions. The diversity in the dataset was crucial for training a robust model capable of handling real-world variances.

- **Traning Strategy**: The model was trained for 10 epochs with batches of 24, which was determined after multiple tests with different sizes of batches to evaluate what would best suit the specific hardware used for the training 9. After each epoch the data was scrambled before being divided into new batches for the next epoch, this is a common strategy to improve generalisation. The learning rate was set to an initial value of 0.001, adjusted dynamically based on the validation loss.

- **Training Metrics**: The model's training performance was evaluated using metrics usually used for multi-class models like the one here, including accuracy, precision, recall, and F1-score for each class. The loss function was also monitored for the entirety of the training.

5.3.2 Performance Inside Simulation

The model's performance was further evaluated within the simulation environment, where it processed live video feeds from the drone's camera to detect workers' safety compliance and non-compliance.

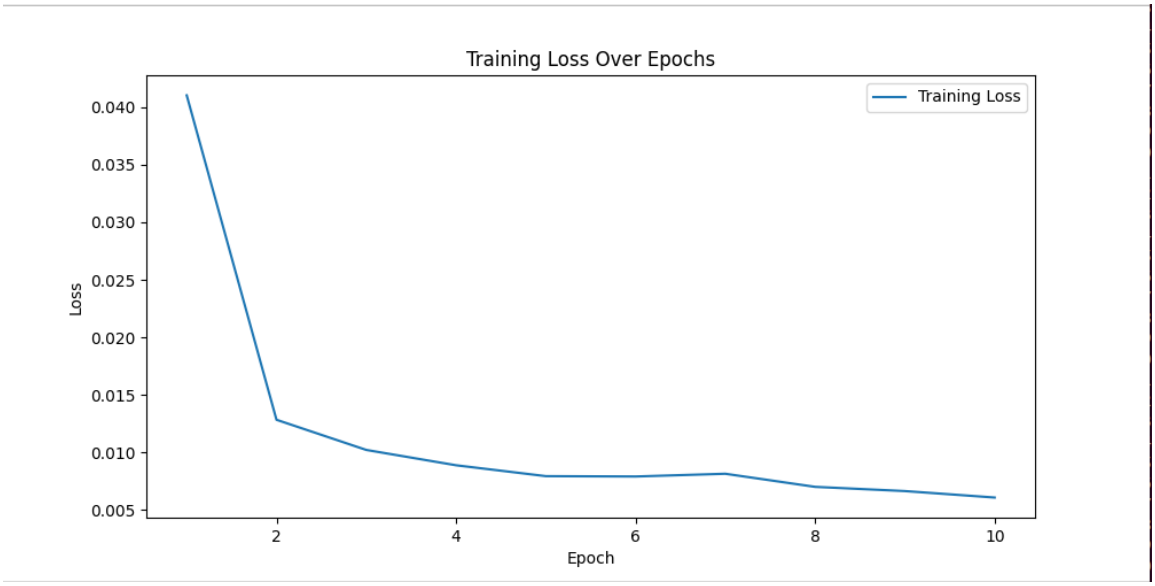


Figure 15: Loss during the 10 epochs of training

	Pred. Non-Com.	**Pred. Com.**	**Pred. Empty**	**Total**
Act. Non-Com.	475	24	1	500
Act. Com.	18	480	2	500
Act. Empty	2	8	90	100
Total	495	512	93	1100

Table 3: Confusion Matrix for training performance

The performance inside the warehouse was measured differently than the training performance, the reason being that when a model is in production there are additional metrics to consider: what is the speed of detection (ms), how many images per second can the model treat (FPS) or what is the performance impact of an increasing number of objects to detect on screen.

- **Detection Speed**: The model maintained a real-time detection speed, processing frames in 76.92ms on average, meaning at approximately 13 frames per second (FPS). However, this number seemed surprisingly low at first, which led to further investigation into why the model was having such poor performance and the discovery was made that the model was actually limited by the simulation running being too slow at sending images through. Therefore, the model was run on a pre-recorded video stream while the simulation was running, to simulate a real load on the system, to fairly test the model's performance without Isaac Sim's video stream bottlenecking the model's performance. This led to a significant improvement with a final delay of 15.82ms per image (63.2 FPS). This is faster than the amount of frames per second most small cameras, similar to the ones found on drones, can record at, they are usually 30FPS or 60FPS. Therefore it can be concluded that the model is fast enough to achieve real-time performance on small-scale cameras.
- **Accuracy**: To measure the accuracy inside the simulation a different strategy needed to be adopted to the synthetic dataset. A series of 40 pre-recorded videos where half contained workers with safety on and the other half safety off. For each video, a CSV file was exported with the detection for each frame, and on average the model correctly identified the type compliance/non-compliance in 74.2% of the images. The reason for this much lower score than the 94.6% on the dataset is the model struggling with half bodies of workers, meaning when the worker is starting to get out of the frame and only a portion of his body is visible the model often wrongly identifies it. An example of this can be seen in figure 16 where the worker is

mostly off the frame and is without a vest but the model detects it as safety compliant, the opposite also happens. This is probably caused by a dataset containing pictures that were too centred and not enough cropped images of workers, therefore, less generalisation and struggles when the full data is not available.

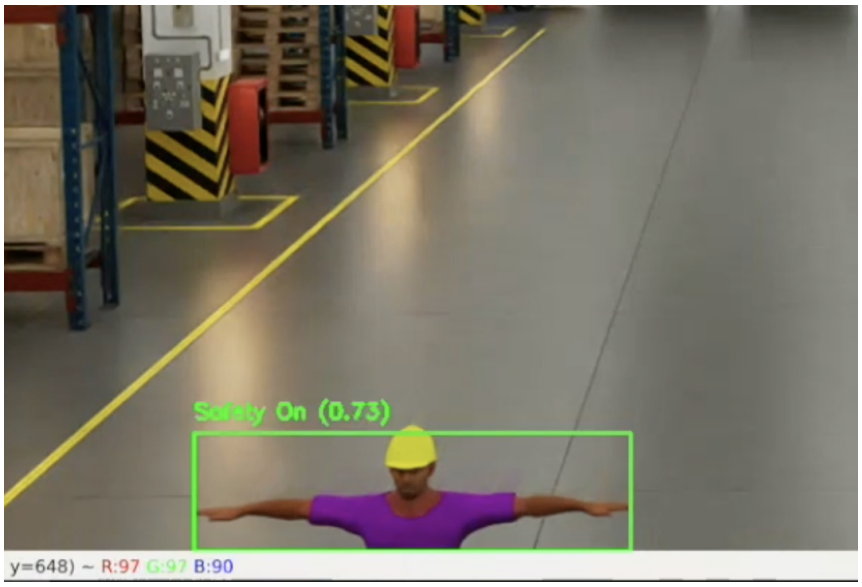


Figure 16: Wrong labelling of cropped worker

- **Miss Detecting Objects**: During the testing on the 40 pre-recorded videos, an unexpected situation occurred where the data showed the detection of a worker on frames where there was not a single worker visible, looking closer at the data and the labelling boxes for each images revealed that the model was repeatedly identifying the fire alarms as workers and assigning them compliant/non-compliant classes. The theory as to why the model kept on wrongfully identifying fire alarms as workers seems to come down to the colour of the worker's shirt, used in the dataset, being similar to the fire alarm's colour. This again reinforces the idea that the dataset needs to be more varied with images of different workers with completely different colours and lighting effects.

- **Performance Impact with Multiple Workers**: In an environment as busy as a warehouse, it is primordial to assess the performance of the Resnet50 model in an environment with many points of interest. The metric used to assess the impact of the increase in point of interest is the FPS and using a similar technique as in the real-world accuracy section above with 2 video streams, one containing a single worker and the second containing 25, the difference in FPS was 15.7% lower in the case with more point of interest. However, according to previous research [17], there is a non-linear scaling in the FPS decrease when increasing the point of interest. This non-linear scaling propriety can have perverse effects on the scaling of this method, however, even in busy warehouses, the amount of workers appearing on the camera feed is unlikely to exceed 25, therefore, while still being considered, it should not affect the method here.

- **Scalability**: Even if this framework was mainly designed with a mono-drone strategy in mind, seeing how it performs with multiple drones could still be appreciated by researchers looking into developing strategies with swarms of drones. Unfortunately, scaling the strategy from one to two drones led to multiple crashes and it was unable to fully run the simulation. However, using the simplified virtual environment 5.1 and modifying the parameters of the drones to lower the definitions of their cameras from 1080 x 1920 to 720 x 1280, a steady 11.2 FPS average was maintained by the simulation environment with two drones running object detection and the navigation system. Therefore, With faster hardware, better optimisation settings and a lighter computer vision model, a simulation with 4 drones could be in the realm of possibilities.

5.3.3 Performance on Real-World Data

The feasibility of the Sim-to-Real strategy was tested by evaluating the model's performance on real-world data collected from actual warehouse operations. The strategy used to determine how well the model performs on real images of workers was to utilise a readily available dataset of workers inside warehouses [18].

- **Detection Accuracy**: The results of this test were not conclusive at all, the model performed extremely poorly on real-world images, predicting workers on images where there were none, not detecting workers on others and wrongly labelling compliant/non-compliant workers. The model achieved an accuracy rate of approximately 43% in real-world conditions, which is extremely poor compared to the 95% on the training dataset. This suggests again that the dataset used to train the model is not diverse enough and overfits the particular worker's 3D model inside Isaac Sim. However, those results cannot be fully discredited since on some occasions the model accurately detected workers as being compliant or not compliant, which suggests that the Sim-to-Real strategy could perform well with a higher quality and more diverse dataset.

	Pred. Random	**Pred. Vest**	**Pred. No Vest**	**Total**
Actual Random	95	115	90	300
Actual Vest	120	145	85	350
Actual No Vest	85	110	155	350
Total	300	370	330	1000

Table 4: Confusion Matrix Reflecting Poor Performance of ResNet-50 on real-world Hardhat and Safety Vest Dataset (Randomised)

Figure 17: Example of real-world data correctly recognising safety compliance

As stated earlier, the confusion table for real-world data does not tell the entire story, since on real data that has similar characteristics to the data generated inside the simulation the model is still able to correctly recognise safety compliance, as demonstrated in figure 17 where the model correctly identifies the worker as safety compliant with a green bounding box around him. This is extremely promising for the future development of this technique, particularly in scenarios where real-world data is difficult to obtain. The ability to generate and easily label large datasets from realistic 3D simulations holds great potential for more efficient and faster deployment of complex Computer vision strategies.

5.3.4 Conclusion of Computer vision model performance

The model, while not perfect, performed extremely well on training data and in the simulation environment, however, as seen previously, it did not perform well on real images of workers, which is likely due to the fact that only a single 3D asset of the worker was used and the texture/lighting parameters were not diverse enough. However, there have been some promising results considering the fact that the neural network was still able to properly detect compliant and non-compliant workers on real images (17), this highlights the potential of Sim-to-Real with Isaac Sim moving forward. A summary of the neural network's performance across training and real-world scenarios has been compiled in table 5.

Metric	**Training**	**Real-World data**
True Positives (TP)	93.67%	39.13%
False Positives (FP)	3.33%	30.42%
True Negatives (TN)	96%	39.94%
False Negatives (FN)	5%	55.71%
AUC-ROC	0.95	0.43
Processing Time per Frame	0.033s	0.048s

Table 5: Performance Metrics Summary

5.4 Environmental Impact

The environmental impact of this project is multifaceted, considering the benefits of using simulation environments, the sustainability of the technology involved, and the overall carbon footprint reduction when compared to traditional real-world testing.

5.4.1 Simulation vs. Real-World Testing

One of the most significant environmental benefits of this project is the reliance on simulation environments for the development and testing of autonomous drone-based systems. By using the Nvidia Omniverse and Isaac Sim for simulations, the project substantially reduces the need for physical prototypes and real-world testing. This approach minimises the consumption of physical resources, such as materials for drone construction and the energy required for repeated test flights. Additionally, simulation environments do not produce waste or emissions associated with traditional testing methods, such as fuel consumption or wear and tear on drone components.

5.4.2 Sustainability of Drone Components

Drones, particularly the ones used in industrial settings like warehouses, are typically constructed with lightweight materials such as plastic and aluminum. The sustainability of these materials is a critical consideration. The project emphasises the importance of using durable, long-lasting components to reduce the frequency of replacements, which in turn minimises waste.

5.4.3 CO2 Impact Comparison: Small-Scale Drone vs. Spot Robot

To estimate the CO2 emissions from constructing a small-scale drone and Boston Dynamics' Spot robot, we base the calculations on the average weight of each and the typical CO2 emissions for electronics manufacturing.

According to recent estimates, electronics manufacturing emits approximately **30 kg of CO2 per kilogram of electronic equipment** [19], considering this let's compare the carbon footprint of drones and the most popular warehouse monitoring robot from Boston Dynamics:

- **Small-Scale Drone** (e.g., DJI Phantom 4):

$$\text{CO2 Impact}_{\text{Drone}} = 1.5\,\text{kg} \times 30\,\text{kg CO2 per kg} = 45\,\text{kg CO2}$$

- **Spot Robot** (Boston Dynamics Spot):

$$\text{CO2 Impact}_{\text{Spot}} = 32.5\,\text{kg} \times 30\,\text{kg CO2 per kg} = 975\,\text{kg CO2}$$

According to the estimation, drones would reduce the carbon impact of robotised monitoring systems in warehouses by approximately 95.38%, which is a very significant reduction. This is another argument towards making drones a more viable option for warehouse monitoring.

6 Evaluation and Impact

6.1 Reflecting on Results

Reflecting on all the results and data produced throughout this framework's evaluation is key to analysing the future steps necessary to improve the project.

6.1.1 Simulations Performance

After completing this project, it has become very apparent how important the performance of the simulation (FPS, memory load, CPU load...) is to make the framework more usable and accessible. Therefore, having an extensive result section to analyse the performance optimisation is key.

Throughout different experimentations with settings and parameters, the minimum spec requirement for a computer to run the simulation was drastically lowered, from expensive professional-grade hardware to lower-cost consumer-grade hardware. This aligns with the goal of this framework: Making the the research for drone-based monitoring systems much more accessible.

6.1.2 Drone Control

The results on drone control were mixed, the solution provided a very usable baseline navigation system with customisable parameters, in the form of weights between collision avoidance and navigation system, but the maximum rate for hitting the target was only around 70%, which is a promising start but much lower than anything that could be accepted in the real world. That would mean the drone would fail on nearly a third of its missions and would need human intervention to unstuck it from its endless loop (about 10% of missions) or crash and risk injuring people in 20% of occasions, which would be completely unacceptable on real-life implementations of a similar system.

However, this strategy still allows for an easily usable starting point navigation to test strategies and provides a baseline for the research community to improve on.

6.1.3 Computer Vision

The Resnet50 model performed very well on the test dataset during training and reasonably well inside the simulation, however, it performed extremely poorly on real-world data. The common factor of the relatively poor performance does not seem to stand from the choice of the model or the Faster R-CNN technique but rather from the generation of the dataset. The images used to train the model were not varied enough, the same worker model was used for the entire dataset, the textures and colours, lighting effects and background were not varied enough throughout the generation of the images and images of partially visible workers were not included. Therefore, the model seemed like it overfitted the training data, hence why it performed so well on test data but did not generalise enough to be flexible throughout different scenarios.

Despite these relatively poor data generation techniques, the model was still able to recognise a real pictures of compliant and non-compliant workers, which is extremely promising for the usability of Sim-to-Real techniques.

6.1.4 Sustainability

The findings regarding sustainability were extremely promising, with an estimated reduction in Co2 of approximately 95%. However, these are relatively simple estimations and do not encompass the entire complexity of such problems, for example, smaller-scale robots like drones have small batteries, which tend to degrade a lot faster than larger batteries, like the ones in the Boston Dynamic robots. This could lead to significantly more battery replacement with drones and an increased impact on E-Waste production. Furthermore, drones are less sturdy and more likely to break upon collision, which could lead to an increase in carbon footprint due to the replacement of drones.

6.2 Future Work

In this section, clear goals to improve the current solution will be formulated based on the previous analysis of the final results and the limitations of the framework.

6.2.1 Simulations Performance

Regarding the performance of the simulation, the results are excellent all-around, however, when pushing the framework beyond what it was designed for to use it for swarms of drones, the performance is relatively poor. The main contributor to these lackluster results is the overhead in rendering for multiple cameras, meaning each drone needs to render its own POV and that is extremely taxing on the GPU. To combat that, a potential area of improvement could be to offer a modified navigation and computer vision system where the cameras would not be very low definition. Since the Resnet model accepts as inputs 224x224 pixel images, rendering the scene in full HD (1080x1920 pixels) could be seen as a wasted performance that could be allocated instead to more cameras of lower resolution and allow for swarms of drones to benefit from this framework. However, the impact of reducing the resolution of the cameras on the Computer Vision model needs to be carefully studied before implementation.

6.2.2 Drone Control

The navigation strategy yielded very unsatisfactory results with a high crash rate and a relatively low success rate. The analysis of the results (6.1.2) suggests that the hybrid system of weight balance between hard-coded GPS location and crash avoidance might be inherently flawed. This indicates that 2 possible options are to consider, either to only use very precise hard-coded aerial pathways/routes based on GPS location which would be much easier to implement and eliminate the situations where the drone would get stuck in an endless collision avoidance loop. However, this would make the strategy a lot less flexible and adaptable which is not ideal in a dynamic environment. The second option would be to use a fully autonomous path-finding system, like the ones discussed in the literature review, which would make the drones much more flexible and adaptable. However, this comes with massively increased development and testing time while still not completely ruling out the possibility of the drone wrongly interpreting a situation and making a poor decision that leads to a crash and potential injuries.

It seems like the choice of which type of strategy to implement in this case comes down to compromises and trade-offs, there does not seem to be an all-around perfect solution. It will depends on the needs of the researchers and careful analysis of the performance of each implementation.

6.2.3 Computer Vision

The performance problems of the Resnet50 model seem to indicate that the dataset was not diverse enough. To combat this problem a list of improvements is needed: more diverse textures (different coloured vests, trousers, teeshirts), heterogenous backgrounds, different lighting angles, partial visibility of the subject, and varied joint positions for the different limbs. Putting all these parameters

in place is suspected to have a drastic improvement in the model's capabilities to generalise. Additionally, including a small subset of the dataset as real-world images might also significantly improve the model's generalisation.

6.2.4 Sustainability

In regards to the sustainability results of the project, further in-depth analysis of the Co2 impact of drones compared to existing solutions is needed. Other metrics to consider in this analysis are the impact of running the simulation (electricity, heat...), the potential future e-waste due to the hardware required to run this framework, the impact of increased battery replacement rates and damage rates.

7 Legal, Social, Ethical and Professional Issues

The development and deployment of autonomous drone systems for warehouse safety audits raises important legal, social, ethical and professional considerations that must be carefully addressed:

7.1 Legal and Regulatory Considerations

7.1.1 Compliance and Privacy

The use of drones in indoor commercial spaces like warehouses falls under various regulatory frameworks depending on the jurisdiction. Operators need to comply with workplace safety regulations and rules around unmanned aerial vehicles in industrial settings. Additionally, the system will capture video/image data of workers, necessitating compliance with data protection regulations like GDPR in Europe or various state privacy laws in the US. Key considerations include obtaining proper consent, implementing data minimisation practices, ensuring secure data handling, and establishing clear retention and deletion policies.

7.1.2 Liability and Intellectual Property

In case of accidents or incidents involving the drone system, clear frameworks need to be established to determine responsibility between system developers, warehouse operators, and workers. The complex nature of the system also raises IP considerations, requiring proper licensing and attribution for third-party components and potentially seeking patent protection for novel innovations.

7.2 Social and Cultural Impact

7.2.1 Workforce Dynamics

While the system aims to enhance safety, concerns may arise about potential job displacement of human safety inspectors. Clear communication about the system's role in augmenting rather than replacing human workers is crucial. The introduction of autonomous drones may also face resistance from workers, requiring efforts to socialise the concept and demonstrate its benefits. Trust in the technology's capabilities and decisions is essential for its effectiveness.

7.2.2 Equality and Cultural Considerations

The implementation of advanced technological solutions may exacerbate existing digital divides within the workforce, potentially leading to inequalities. In multinational companies, the acceptance and effectiveness of such systems may vary across different cultural contexts, with surveillance norms differing between cultures.

7.3 Ethical Implications

7.3.1 Autonomy, Fairness, and Transparency

While the system is autonomous, decisions about intervention based on its findings raise ethical questions. Clear protocols for human oversight are necessary. The AI models must be rigorously tested to prevent biases based on factors like race, gender, or age. The decision-making process should be as transparent as possible, allowing workers to understand and contest decisions if necessary.

7.3.2 Privacy, Accountability, and Dual Use

Constant surveillance, even for safety purposes, can impinge on workers' privacy and dignity. The system should balance safety needs with respect for individual privacy. Clear accountability mechanisms are needed for system errors or failures. While designed for safety audits, the technology could potentially be repurposed for more invasive surveillance, necessitating ethical guidelines to prevent misuse.

7.4 Professional Responsibilities

7.4.1 Competence and Collaboration

Developers and operators must ensure they have the necessary skills to develop and maintain the system responsibly, committing to continuous learning given rapid technological advancements. The complex nature of the system requires interdisciplinary collaboration across robotics, AI, law, ethics, and social sciences.

7.4.2 Ethical Practice and Stakeholder Engagement

Professionals should adhere to relevant codes of ethics, such as those provided by IEEE or ACM. They should engage with all stakeholders including workers, managers, safety officers, and regulators to ensure the system meets diverse needs and concerns. Maintaining detailed documentation, conducting thorough risk assessments, and engaging in public communication about the system's benefits and limitations are crucial professional responsibilities.

Addressing these legal, social, ethical, and professional issues is crucial for the responsible development and deployment of autonomous drone systems in warehouses. It requires ongoing dialogue, iterative improvement, and a commitment to balancing technological innovation with human values and societal norms.

8 Conclusion

The development and deployment of autonomous drone systems for warehouse safety audits represent a significant step forward in the integration of advanced robotics into industrial operations. This project successfully demonstrated the viability of a Sim-to-Real approach, utilising the Pegasus Simulator within Nvidia's Omniverse and Isaac Sim environments to create a highly realistic and functional virtual warehouse. Through rigorous testing and refinement, the system has shown its potential to enhance safety, streamline operations, and reduce the reliance on human intervention in hazardous or routine inspection tasks.

By leveraging existing frameworks like ROS and Mavlink, the project was able to integrate proven communication protocols that facilitate both high-level and low-level control of drone operations. This flexibility is crucial for adapting the system to various real-world scenarios, ensuring that it can meet the diverse needs of modern warehouses.

The outcomes of this project underscore the importance of simulation in the development of autonomous systems, offering a cost-effective and scalable method for testing and optimising drone strategies before real-world deployment. The successful application of Sim-to-Real methodologies

not only validates the approach but also opens new avenues for future research and development in the field of industrial automation.

Additionally, the shortcomings of each technique have been fully assessed and guidance for future work has also been detailed. This will help move the framework forward into a true fully implemented and robust solution.

As with any emerging technology, the implementation of autonomous drones in warehouses brings with it a host of legal, social, ethical, and professional considerations. This project has highlighted the need for ongoing dialogue and careful consideration of these issues to ensure that the deployment of such systems is both responsible and beneficial to all stakeholders involved.

In conclusion, the advancements achieved through this project provide a solid foundation for the continued exploration and enhancement of autonomous drone systems in industrial environments. By balancing technological innovation with a commitment to ethical practices, we can look forward to a future where drones play an integral role in maintaining safe, efficient, and resilient industrial operations.

9 Appendix

- The full code with guidance on how to run it can be found here: https://github.com/William-Droin/Sim-to-Real-Framework-for-Autonomous-Drone-Based-System-in-Warehouses

Table 6: Technical Specifications of the Computer System

Component	**Specification**
Processor (CPU)	AMD Ryzen 9 3900X
	12 Cores, 24 Threads Base Clock: 3.8 GHz Max Boost Clock: 4.6 GHz L3 Cache: 64 MB TDP: 105W
Memory (RAM)	32 GB DDR4
	Speed: 3200 MHz Type: Dual-channel Form Factor: DIMM
Storage	1 TB NVMe SSD PCIe 4.0
	Read Speed: Up to 5000 MB/s Write Speed: Up to 4400 MB/s Form Factor: M.2 2280
Graphics Card (GPU)	NVIDIA GeForce RTX 4060 Ti
	Memory: 16 GB GDDR6 CUDA Cores: 4352 Boost Clock: Up to 2535 MHz TDP: 160W Interface: PCIe 4.0 x16
Motherboard	Compatible with PCIe 4.0, AM4 Socket
	Supports DDR4 RAM NVMe PCIe 4.0 M.2 Slot USB 3.2 Gen 2 Ports
Power Supply Unit (PSU)	650W 80+ Gold Certified
Cooling System	Air Cooling (Cooler Master Hyper 212 or equivalent)
Operating System	Windows 10/11 or Linux (User's Choice)

```
1 import omni.replicator.core as rep
```

```python

prop_position = (-47.94876, 35.56706, 0)

with rep.new_layer():

    # ============================
    # Import the worker's 3D model
    # ============================
    camera = rep.create.camera()
    render_product = rep.create.render_product(camera, (1000, 1000))


    # ============================
    # Import the worker's 3D model
    # ============================
    worker = rep.create.from_usd(f"omniverse://localhost/NVIDIA/Assets/Isaac/2023.1.1/Isaac/People/Characters/male_adult_construction_05_new/male_adult_construction_05_new.usd", semantics=[('class', 'box')])
    rep.modify.pose(input_prims=worker, rotation=(90,0,0), position=prop_position)


    # ============================
    # Set up the Writer that is responsible for saving and labelling the images
    # ============================
    writer = rep.WriterRegistry.get("BasicWriter")
    writer.initialize(output_dir="_output", rgb=True, bounding_box_2d_tight=True)
    writer.attach([render_product])


    #============================
    # Trigger the modification of the camera angle on each frame for the 10 000 following frames
    #============================
    with rep.trigger.on_frame(num_frames=10000):
        with camera:
            rep.modify.pose(rep.distribution.uniform((prop_position[0]-6, prop_position[1]-6, 3), (prop_position[0]+6, prop_position[1]+6, 3)), look_at=worker)

    rep.orchestrator.preview()
```

Code 1: Python Replicator Code

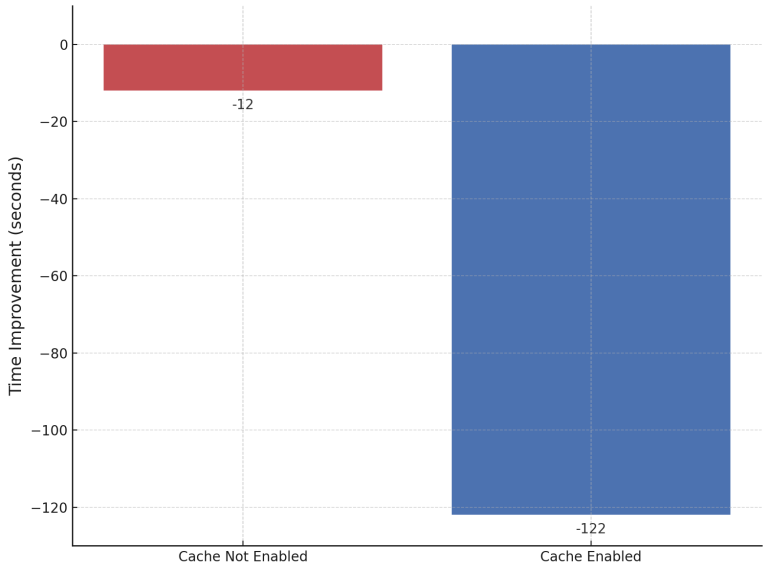


Figure 18: Memory Utilisation (lower is better)

References

[1] A. A. Tubis, J. Ryczyński, and A. Żurek, Risk assessment for the use of drones in warehouse operations in the first phase of introducing the service to the market *Sensors*, vol. 21, no. 20, p. 6713, 2021.

[2] D. Büchler, S. Tebbe, *et al.*, i-Sim2Real: Reinforcement Learning of Robotic Policies in Tight Human-Robot Interaction Loops *arXiv preprint arXiv:2207.06572*, 2023.

[3] M. Gubán and J. Udvaros, A Path Planning Model with a Genetic Algorithm for Stock Inventory Using a Swarm of Drones *Drones*, vol. 6, no. 11, 2022.

[4] T. Silver, N. J. Kumar, S. Proulx, J. Barry, L. Zhao, W. McClinton, L. P. Kaelbling, and T. Lozano-Pérez, Practice Makes Perfect: Planning to Learn Skill Parameter Policies *arXiv preprint arXiv:2402.05678*, 2024. Collaboration between MIT CSAIL and Boston Dynamics.

[5] B. Xu, F. Gao, C. Yu, R. Zhang, Y. Wu, and Y. Wang, OmniDrones: An Efficient and Flexible Platform for Reinforcement Learning in Drone Control 2023.

[6] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White, Sim2Real in Robotics and Automation: Applications and Challenges *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[7] B. Rahmadya, R. Sun, S. Takeda, K. Kagoshima, and M. Umehira, A framework to determine secure distances for either drones or robots based inventory management systems *IEEE Access*, vol. 8, pp. 170153–170161, 2020.

[8] F. B. Sorbelli, F. Corò, C. M. Pinotti, and A. Shende, Automated picking system employing a drone in *Proceedings of the 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 633–640, IEEE, 2019.

[9] Y. Wu and K. H. Low, An Adaptive Path Replanning Method for Coordinated Operations of Drone in Dynamic Urban Environments *IEEE Systems Journal*, vol. 15, no. 3, pp. 4600–4611, 2021.

[10] A. Serrano-Munoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, skrl: Modular and flexible library for reinforcement learning *Journal of Machine Learning Research*, vol. 24, no. 254, pp. 1–9, 2023.

[11] E. Chaffre *et al.*, Sim2Real Transfer in Deep Reinforcement Learning for Robotics: A Survey *arXiv preprint arXiv:2009.13303*, 2021.

[12] J. Tobin *et al.*, Domain randomization for transferring deep neural networks from simulation to the real world in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.

[13] I. Akkaya, M. Andrychowicz, *et al.*, Solving Rubik's Cube with a Robot Hand *arXiv preprint arXiv:1910.07113*, 2019.

[14] M. L. Hoang, Smart Drone Surveillance System Based on AI and on IoT Communication in Case of Intrusion and Fire Accident *Drones*, vol. 7, no. 12, 2023.

[15] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, Imagenet: A large-scale hierarchical image database *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[17] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, Speed/accuracy trade-offs for modern convolutional object detectors in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7310–7311, 2017.

[18] J. Yin, Hardhat and Safety Vest Image for Object Detection 2021. Accessed: 2024-08-14.

[19] J. Malmodin and P. Lundén, The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015 *Sustainability*, vol. 10, no. 9, p. 3027, 2018.