Osprey Simulator

A Simulator Framework for Fixed-Wing UAV Updraft Localization

Master Thesis

Kjell Vleeschouwer

Delft University of Technology

TU Delft

Osprey Simulator

A Simulator Framework for Fixed-Wing UAV Updraft Localization

by

Kjell Vleeschouwer

Master Thesis to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday February 5, 2025 at 9:30 AM.

Thesis committee:

Chair:	Dr. E. J.J. Smeur	TU Delft, AE
Supervisors:	Prof. Dr. G. C. H. E. de Croon	TU Delft, AE
	Ir. B. D. W. Remes	TU Delft, AE
	Ir. S. Hwang	TU Delft, AE
External examiner:	Dr. R.T. Rajan	TU Delft, EEMCS
Student number:	4557107	
Project duration:	June 1, 2023 – December 12, 2024	

Cover: Image captured from a simulation within the Osprey Simulator

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

I would like to begin by expressing my sincere gratitude to my supervisors, Guido, Sunyou, and Bart, for their guidance and unwavering support during the final stages of my studies at TU Delft. Over the past year and a half, I have navigated many challenges, balancing the demanding final phase of my thesis with a full-time job, a job for which I needed the software skills I developed during this project. The flexibility and understanding shown by my supervisors have been invaluable, allowing me to close the TU Delft chapter with both a project I am proud of and the opportunity to start a job that embodies the saying, “love what you do and you’ll never work a day in your life.” I also hope that someone at the MAVLab can build further on this simulator framework, ultimately developing a real-life test for soaring detection by first training it with the simulator.

I would also like to extend my heartfelt thanks to my girlfriend, Emma. Her unconditional support, timely warnings when procrastination loomed, and her own inspiring academic dedication have been crucial to my progress. Similarly, I am grateful to my sister Britt, who’s motivational speeches and own recent achievement of becoming a surgeon stands as a testament to hard work and determination, a benchmark to which I continue to aspire.

Lastly, I offer my deepest thanks to my parents for their endless love, encouragement, and the opportunities they have provided me throughout my life. Their support has been the foundation upon which I have built my present and continue to shape my future. Graduating from TU Delft marks a significant personal milestone, yet it is also a tribute to the steadfast support they have given me throughout my life. I am forever grateful.

Kjell Vleeschouwer
Delft, February 2025

Contents

List of Figures

List of Tables

Abbreviations

AP	Average Precision	35, 37
API	application programming interface	49
CFD	Computational Fluid Dynamics	15, 23–25, 41–45, 56
CNNs	Convolutional Neural Networks	34
DES	Detached Eddy Simulation	41
DL	Deep Learning	33
DRL	Deep Reinforcement Learning	32, 33
GANs	Generative Adversarial Networks	38
INDI	Incremental Nonlinear Dynamic Inversion	23
IoU	Intersection over Union	55
LES	Large Eddy Simulation	41, 42
LLM's	Large Language Models	33
MAE	Mean Absolute Error	55
MAV's	Micro Aerial Vehicles	24
ML	Machine Learning	15, 22, 31, 40, 42, 44, 47, 56
NLP	Natural Language Processing	33
RANS	Reynolds Averaged Navier Stokes	41
RL	Reinforcement Learning	22, 23, 31–33
RMSE	Root Mean Squared Error	55
SSL	Self-Supervised Learning	38, 39, 42, 47
UAV	Unmanned Aerial Vehicle	17, 18, 47
UAV's	Unmanned Aerial Vehicles	15, 17, 20–25, 27, 29–33, 36, 38–40, 43, 44, 47
USD	Universal Scene Description	iii, 49, 50
VTOL	Vertical Takeoff and Landing	21

Part I

Scientific Paper

Osprey Simulator: A simulator framework for fixed-wing UAV updraft localization

Kjell Vleeschouwer , Sunyou Hwang*, Bart D.W. Remes*, Guido C.H.E. de Croon*

***Abstract*—The real-world application of Micro Air Vehicles (MAVs) is often constrained by their limited flight range and endurance, primarily due to battery limitations. One way to overcome this challenge is by leveraging naturally occurring vertical air currents, a technique inspired by soaring birds. This paper introduces Osprey Simulator, a framework designed to simulate and test energy-efficient soaring flight strategies for fixed-wing drones. Built on Isaac Sim, Osprey Simulator enables the creation of both randomly generated urban environments and real-world environments, such as the TU Delft campus. These environments are paired with accurate wind field simulations using OpenFOAM, providing a robust platform for studying aerodynamic interactions in diverse settings. This functionality allows for the generation of scalable synthetic datasets, including depth images and wind field information, enabling comprehensive exploration of soaring potential across various structural geometries and wind conditions. Using generated synthetic data, a neural network was trained to predict optimal soaring regions by analyzing depth images and wind field information. The network demonstrates the ability to identify updraft and downdraft regions, enabling more efficient path planning for drones in urban environments. By integrating realistic simulations and advanced predictive models, Osprey Simulator serves as a powerful tool for advancing autonomous soaring and extending the operational range of fixed-wing MAVs.**

Supplementary Material

Code: https://github.com/kv8-A/OspreySimulator

I. Introduction

Small flying robots, commonly referred to as Micro Air Vehicles (MAVs), are increasingly used in a variety of real-world applications, from windmill inspection and agricultural surveying to search and rescue operations. However, a major limitation of MAVs lies in their restricted flight range and endurance, stemming from constraints on size, weight and mainly their battery. One way to extend their operational duration and coverage is to harness energy form the environment. When looking at nature, birds achieve this through soaring, which is the ability to get energy out of the environment by harnessing naturally occurring spots of vertical air currents. These currents can come in the form of thermal updrafts, orographic updrafts or or dynamic soaring which makes use of wind gradients [1][2].

In recent years, researchers have begun to investigate how MAVs and UAVs (Unmanned Aerial Vechicles) can mimic these soaring techniques to improve the endurance of MAVs. While thermal updrafts, rising columns of warm air, have long been exploited by gliders, MAVs can also utilize orographic lift, which arises when wind is deflected upward by terrain or structures.UAVs can also utilize orographic lift, which is created when wind is deflected upward by terrain or structures. These orographic soaring techniques have been researched and tested in real life, and autonomous soaring controllers have even been developed [3][4][5]. More recently, autonomous soaring has also been demonstrated in [6]. However, these approaches predominantly focus on local control near known soaring locations, requiring the MAV to already be within or very close to an updraft region.

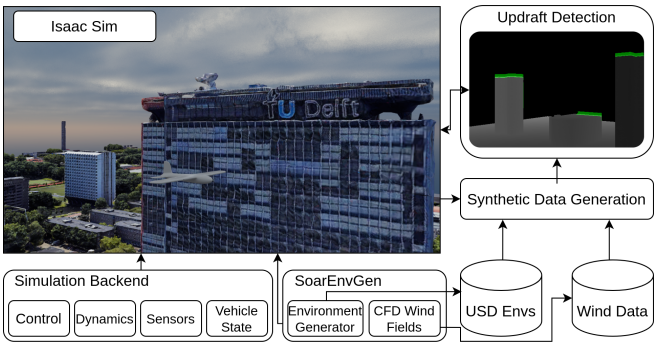


Fig. 1. System Architecture of Osprey Simulator. SoarEnvGen prepares the environments and the wind fields. With the then Synthetically generated data, the updraft recognition networ is created.

For autonomous soaring to become practical in real-world conditions, MAVs must detect and predict soaring opportunities from a distance rather than just react once they are nearby. This requires integrating environmental cues such as vision-based terrain recognition, depth sensing, and sophisticated wind field modeling. By identifying features such as tall buildings or cliffs and understanding the associated wind conditions, MAVs can be guided to optimal soaring spots before arrival. Instead of relying on manually crafted, error-prone algorithms, this paper proposes leveraging deep neural networks to generalize the detection and exploitation of these spots.

To support such research, this paper introduces the Osprey Simulator, a new simulation framework built on top of the Pegasus Simulator and NVIDIA Isaac Sim platforms [7][8]. Developed to advance research on autonomous UAV soaring and orographic updraft detection, The Osprey Simulator automatically integrates realistic wind fields, supports vision and depth sensors, and provides both randomly generated and real-world simulation environments. It expands upon the

*thesis supervisors.

high-fidelity capabilities of Isaac Sim by incorporating complex wind dynamics and fixed-wing UAV models, enabling researchers to develop and test neural networks, like the introduced SoarDetect, for identifying and utilizing orographic soaring regions. his platform paves the way for robust, ultimately advancing MAV autonomy and endurance.

The remainder of this paper is organized as follows: Section II discusses related work in more detail. Section III presents the Osprey Simulator and its core functionalities. Section IV details the use of synthetic self-supervised data to set up and train a neural network for updraft detection. Finally, Section V concludes with a discussion of future research directions.

II. Related Work

The idea of soaring from UAVs is not new and was first introduced in 2005 by Allen. Allen recognized the possibilities for UAVs that could potentially increase their endurance between 2 and 14 hours [1]. The idea has been further explored in [9, 4, 3, 10, 11, 12, 13]. Although the focus mainly has been on thermal soaring and dynamic soaring, these ideas have been explored for orographic soaring in [6] and [14]. In [6], Hwang created an algorithm where autonomous soaring in a wind tunnel was performed, while in [14], de Jong et al. successfully developed a soaring controller on a moving object. This work is used as motivation for Osprey Simulator, as the hope is to use the real-life dynamics and physics to test these controllers in simulation first.

Orographic soaring and the shape of wind fields in urban environments were further explored in [4], [10]. In [4], wind tunnel tests were conducted using a subscale model to the wind fields, and more specifically the possibility for updrafts around buildings. These wind tunnel tests were later validated experimentally by performing tests on the top of the real building. [10] performed a first 3D CFD analysis where the results were in line with what was to be expected from the tests performed in [11] & [4]. The research performed in [12] examined the wind fields within a building complex in Singapore using OpenFOAM for Computational Fluid Dynamics (CFD) with the aim of understanding the wind effects on UAVs' flight paths.

Throughout the years, multiple simulators have been developed for all kinds of robotics and UAV cases, with most of them focused on quadrotor dynamics. However a simulator specifically designed for fixed-wing UAVs that is able to simulate realistic wind fields generated with CFD, seems to be missing. This section will go through the several platforms that have been developed, with each addressing different requirements for detailed, realistic modeling of robotic systems. Platforms like Gazebo have long been favored for their ROS integration & physics engine; however, it lacks the ability to simulate complex visual environments. This gap has been narrowed by the development of simulators like AirSim & Flightmare, which respectively utilize the advanced game engines Unreal Engine and Unity [15] [16][17][18][19].

Building on existing technologies, NVIDIA has introduced its Omniverse suite, which includes Isaac Sim. Isaac Sim offers sophisticated simulation capabilities tailored to robotics applications by making use of NVIDIA's expertise in GPU-accelerated computing and AI. Isaac Sim excels in modal data generation and high-quality RTX rendering. All of this is integrated under the Universal Scene Description standard for complex 3D environment representation. USD has been developed by Pixar. Despite its comprehensive features, Isaac Sim lacks in areas critical for UAV and other aerial vehicle simulation. Examples of the support it lacks include sensors like a barometer, airspeed vector, and more [7] [8].

Trying to fill this gap in aerial simulation within the NVIDIA Omniverse, Pegasus Simulator was developed. Pegasus Simulator has been built upon Isaac Sim. It distinguishes itself by its emphasis on integration with widely adopted frameworks like PX4-autopilot and ROS2, while focusing primarily on quadrotor dynamics. Additionally, it supports both software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations. It also further extends the mission sensors needed for aerial simulation. Comparing it to Osprey Simulator, it still misses two important capabilities: realistic wind fields & fixed-wing dynamics. Building on the foundation of Pegasus Simulator, GSL-Bench has been developed as a sophisticated benchmarking suite tailored specifically for the field of Gas Source Localization (GSL) [20]. GSL-Bench addresses the challenges of GSL by integrating the high-fidelity graphics powered by Isaac Sim and the creation of random environment generation on which CFD simulation is performed. Although these environments consist of indoor warehouses and others, GSL-Bench is one of the first simulators integrating simulation with high-fidelity wind fields inside the simulator, something that has been performed for Osprey Simulator as well.

Recently, in [21], a software pipeline was developed where they had CFD-generated urban wind fields inside the simulator for quadrotor flight. Here, the simulation integrates OpenStreetMap for real-world building geometry, OpenFOAM for CFD and wind field computations, and the Gazebo simulation environment. The software pipeline imports these urban geometries and performs the CFD analysis using the steady-state RANS equation with a k-ϵ turbulence model. After the CFD generation, a customized Gazebo plugin interpolates the generated CFD wind field at runtime, influencing the quadrotor's dynamics. The results showed the impact of wind on the quadrotor's dynamics and highlighted the impact of urban wind fields on the quadrotor's stability and control.

The approach within the Osprey Simulator resembles the work described above, with key differences in the generation of real-world building geometries as well as the simulation environment used. Unlike the referenced work, the Osprey Simulator supports fixed-wing UAVs and uses a Computational Fluid Dynamics (CFD) software pipeline, called SoarEnv, which integrates urban wind fields into the simulation. More about the CFD module SoarEnv can be found in subsection III-D.

III. Simulator Software Architecture

The Osprey Simulator Software consists of multiple modules with the end-goal in mind of creating an operating fixed-wing dynamics UAV flying in a photo-realistic simulation

environment. It also provides a high level of physics fidelity by providing windfields inside the environment that are created by the SoarEnv module within the simulator. The SoarEnv module acts as a Python OpenFOAM automation. Additionally, the SoarEnv module facilitates the generation of realistic wind fields and random urban environments, which are crucial for simulating orographic updrafts. The architecture can be seen in Figure 2. Furthermore, a synthetic data generation module has been developed to create datasets for training neural networks capable of detecting soaring hotspots.

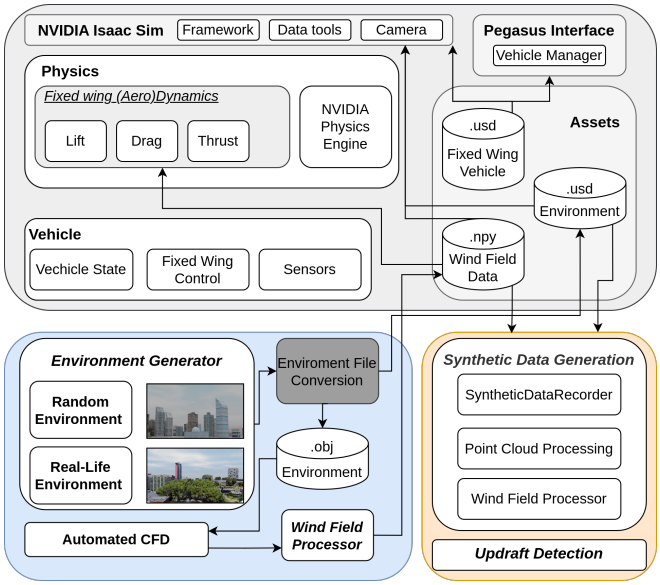


Fig. 2. Software Architecture Schematic

A. NVIDIA ISAAC SIM & Pegasus Simulator

The Osprey simulator is built upon the Pegasus Simulator framework, which is a custom-built extension of NVIDIA Isaac Sim for quadrotor flight.

NVIDIA Isaac Sim offers advanced simulation capabilities tailored to robotics applications, leveraging NVIDIA's expertise in GPU-accelerated computing and AI. However, it lacks support in areas critical for UAV simulation, such as specific sensors and dynamics models.

Pegasus Simulator addresses these gaps by emphasizing quadrotor simulation and integrating widely adopted frameworks like PX4-Autopilot and ROS2, supporting both software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations. It also extends the mission sensors needed for aerial simulation. Nevertheless, Pegasus Simulator still misses two important capabilities: realistic wind fields and fixed-wing dynamics.

Osprey Simulator builds upon Pegasus Simulator by introducing fixed-wing UAV models and integrating realistic wind fields generated through CFD computations via the SoarEnv module. SoarEnv automates OpenFOAM simulations and enables the creation of random urban environments through a custom pipeline in Blender. These features enhance the simulation's realism and enable the testing of fixed-wing UAVs in complex environments with accurate physics. Additionally, the synthetic data generation API within Osprey Simulator allows for the creation of datasets necessary for training neural networks aimed at detecting soaring hotspots, further expanding the simulator's capabilities.

B. Simulation Environment

The simulation environment in Osprey Simulator is a critical component that provides realistic settings for testing fixed-wing UAVs. Environment generation is facilitated through two primary methods, both integrated within the SoarEnv module: importing real-world environments using the Blender-OSM (Blosm) add-on and generating random urban environments using a custom Python module interacting with Blender.[22, 23].

The Blosm add-on for Blender, formerly known as Blender-OSM, significantly enhances environmental simulation capabilities by enabling the integration of OpenStreetMap, Google 3D cities, and real-world terrain data into Blender [22, 23]. Incorporated into SoarEnv, the Blosm add-on allows for the creation of rich, realistic 3D environments by importing accurate geographic data. This streamlines the process of generating landscapes and urban settings that reflect actual locations. An example of such an environment within the Osprey Simulator is shown in Figure 3, showing the TU Delft campus. These environments can serve as exact locations where future testing will be performed, providing the ability to quantify the simulation-to-reality gap.

Fig. 3. The TU Delft Campus

Both methods require conversion of the environment files for compatibility with different components of the simulator. Since both the simulation environment and the CFD API to OpenFOAM do not accept the '.blend' format, the environment files are converted to '.usd' files for Isaac Sim and '.obj' files for the CFD computations. SoarEnv automates this conversion process, ensuring that the scales and coordinates remain consistent across formats. This is crucial because the windfield generated from the '.obj' file must align accurately with the environment in the '.usd' file within the simulator.

The Osprey Simulator provides a seamless workflow for preparing realistic simulation environments. This integration

enhances the simulator's capability to test fixed-wing UAVs in complex urban settings with accurate physics and environmental interactions.

C. Vehicle model

A detailed vehicle model is implemented to replicate the behavior of fixed-wing UAVs. It integrates a 3D drone model via a USD file, defining the UAV's visual and physical characteristics with a single articulation point to streamline dynamics. Configurable parameters, such as mass, aerodynamic coefficients, and propulsion details, allow users to adjust performance attributes, enabling testing of fixed-wing UAVs under diverse conditions.

1) Vehicle & System Dynamics: The Osprey Simulator models fixed-wing UAV dynamics using fundamental forces and moments. Figure 4 illustrates the primary forces acting on the UAV: lift, drag, thrust, and weight, along with the pitch and yaw moments. For simplicity, the roll moment is excluded, with turns modeled exclusively using yaw. While more complex dynamics models exist, this simplified approach provides sufficient fidelity to simulate soaring flight and study updraft exploitation.

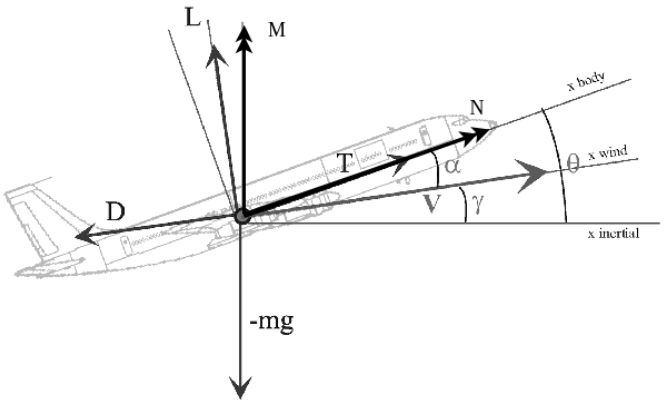


Fig. 4. Fixed-plane dynamics

To model the fixed-wing dynamics, the following equations are employed for lift, drag, and thrust. While straightforward, these equations capture the main aerodynamic effects relevant to the simulator's goal of identifying soaring locations.

Lift is calculated using the lift equation, Equation 1, where the lift coefficient C_L is dependent on the angle of attack α, and the airspeed V accounts for wind interactions.

$$\begin{aligned} L &= \frac{1}{2} \cdot C_L \cdot \rho \cdot V^2 \cdot S \\ C_L &= f(\alpha) \end{aligned} \tag{1}$$

The airspeed V is computed as the sum of the MAV's ground speed and wind velocity components. Both the ground speed and wind velocity are represented as 1D arrays of size 3 in the ground frame of reference. The airspeed used in the equations is the magnitude of the resulting airspeed vector, as the equation is independent of direction (positive or negative):

$$V = \left| \vec{V}_{\text{Ground}} + \vec{V}_{\text{Wind}} \right| \tag{2}$$

Drag is computed similarly to lift, Equation 3, incorporating the drag coefficient C_D, which is dependent on both the lift coefficient and a drag component $C_{D,0}$ that includes components like profile drag [24].

$$D = \frac{1}{2} \cdot C_D \cdot \rho \cdot V^2 \cdot S \tag{3}$$

$$\begin{aligned} C_D &= C_D(C_{D,0}, C_L) \\ C_D &= C_{D,0} + \frac{C_L^2}{\pi \cdot e \cdot AR} \end{aligned} \tag{4}$$

Here, e is the span efficiency factor, and AR is the aspect ratio of the wing.

Thrust is modeled using the equation [24]:

$$T = C_T \cdot \rho \cdot n^2 \cdot D^4 \tag{5}$$

Where

- C_T is the thrust coefficient (assumed constant for this study)
- n is the propeller rpm in revolutions per second
- D is the propeller Diameter
- ρ is the air density

In real-world scenarios, there would be C_T and C_P (power coefficient) curves that vary with the advance ratio, J, defined as [25]:

$$J = \frac{V}{n \cdot D} \tag{6}$$

Here V, is the airspeed. However, for the purposes of this study, the propeller RPM is scaled directly by the power input controlled through the throttle. Both C_T and the influence of the advance ratio on performance are kept constant for simplicity. While this approach deviates from real-life behavior, it is sufficient for capturing the primary effects relevant to fixed-wing UAV dynamics and updraft exploitation.

Although these equations are relatively simple, they adequately represent the primary aerodynamic effects relevant to updraft exploitation. The exclusion of the roll moment, while a limitation, simplifies the model and does not hinder the analysis of soaring flight. Future work could incorporate coordinated turns using roll for increased realism.

By leveraging the wind field data generated by SoarEnv and integrating it with the created aerodynamics, the simulator achieves sufficient fidelity to analyze fixed-wing UAV behavior in diverse wind conditions. This approach enables efficient testing of UAV performance while providing extensive visual and quantitative data for analysis. Furthermore, the simulator is designed to allow future enhancements to its dynamics models, fostering further research in UAV dynamics and autonomous soaring.

2) Sensor Modeling: In addition to the sensor suite already available in Isaac Sim and Pegasus Simulator, several extra sensors are needed for fixed-wing UAVs. These sensors were developed in a way that provides callback functions that can access the UAV state directly. The important sensors included and used inside the Osprey Simulator are:

- Airspeed Sensor: Measures the drone's airspeed by calculating the difference between the ground speed and wind speed. This is essential for aerodynamic calculations and control, particularly in determining lift and drag forces.
- Altimeter: Provides altitude information by measuring the aircraft's height above a reference point, usually sea level. This sensor is crucial for maintaining desired flight levels and for executing altitude-dependent flight maneuvers.
- Angle of Attack Sensor: Measures the angle between the wing chord line and the oncoming airflow (relative wind). This angle is critical for assessing lift generation and for preventing stall conditions.
- Flight Path Sensor: Determines the flight path angle relative to the horizon, helping in trajectory planning and stability control. It provides data on whether the drone is ascending, descending, or flying level. The flight path angle γ can bee seen in Figure 4.
- Heading Indicator: Indicates the drone's heading or directional orientation relative to magnetic north. This sensor aids in navigation and ensures the UAV follows the intended flight path.
- Pitch Angle Gyro: Measures the pitch rate of the drone. This information is vital for attitude control, allowing the UAV to maintain or change its nose-up or nose-down orientation as required.

By integrating these additional sensors, the Osprey Simulator enhances its capability to accurately simulate fixed-wing UAV operations. These sensors provide detailed data necessary for aerodynamic calculations, control algorithm development, and navigation, thereby increasing the fidelity of the simulation.

3) Control with Autopilots: For the purposes of the paper, some minor control was needed to eventually be able to set waypoints to updraft locations, three primary control modes have been implemented. Osprey simulator currently has the altitude hold, velocity hold and heading hold modes available.

- Altitude Hold Controller
  The altitude hold mode is designed to maintain a UAV at a specific altitude using a Proportional-Derivative (PD) controller. The core of the altitude hold system involves computing the altitude error, which is the difference between the desired reference altitude and the current altitude. The PD controller then adjusts the angle of attack to minimize this error. By varying the angle of attack, the UAV can increase or decrease lift to stabilize at the desired altitude. The controller parameters, including the proportional and derivative gains, were fine-tuned to minimize the oscillations and have a short transient response, while also capping the rate of change in the angle of attack to realistic pitch rates. Figure 5 shows the control block diagram.
- Velocity Hold Controller
  The velocity hold mode is responsible for maintaining the UAV's airspeed. Similar to the altitude hold mode, this controller uses a PID approach. The primary input is the difference between the current velocity and the desired reference velocity as well as the current acceleration. The controller this system uses proportional control gains for both acceleration and velocity to regulate throttle adjustments. The controller adjusts the throttle to minimize the velocity error, by changing the throttle setting until the target speed is met. The proportional gains were fine-tuned for a stable response. At the end there is accounted for the range of throttle settings, which is between 0 and 1. The control block diagram can be seen in Figure 6.

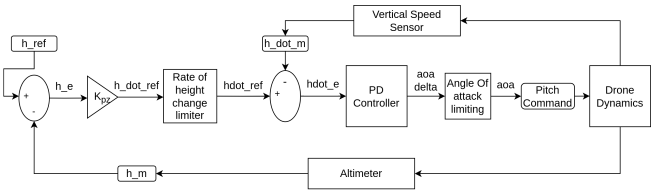


Fig. 5. Block Diagram of The Altitude Hold Controller

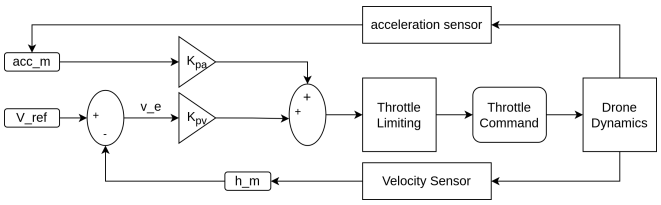


Fig. 6. Velocity Hold Mode schematic

- Heading Hold Controller
  To complete the ability for path following inside the simulator, a heading hold controller was developed as well. Using the same approach as for the altitude & velocity hold controllers, the heading controller was made. Here the controller bases itself on the current heading input and the error with the desired reference heading. The controller calculates the shortest angular distance to the target heading to prevent large angle swings.

D. SoarEnv

SoarEnv is a standalone software pipeline integrated into the larger Osprey Simulator, specifically designed for generating computational fluid dynamics (CFD) wind fields from 3D urban environments. The pipeline operates in three primary stages: first, it converts real-world or randomly generated 3D environments into compatible geometric formats such as .obj or .stl files using a combination of Blender plugins and and a custom Python module. These environments are then processed through OpenFOAM, where steady-state CFD simulations are performed to capture detailed wind behavior around structures. The results of these simulations are exported as wind fields, which are subsequently formatted for use within NVIDIA's Isaac Sim, enabling realistic airflow integration into simulation environments. For a visual overview of this process, this can be seen in Figure 7.

A big factor of this pipeline is that it is highly versatile, capable of generating wind fields for both real-world environments and randomly generated urban layouts. The leveraging

of these precomputed wind fields in SoarEnv allows for the simulations to perform accurate dynamic simulations within the virtual environment, making it an essential tool within the Osprey simulator and other simulations/simulators where accurate wind fields are critical.

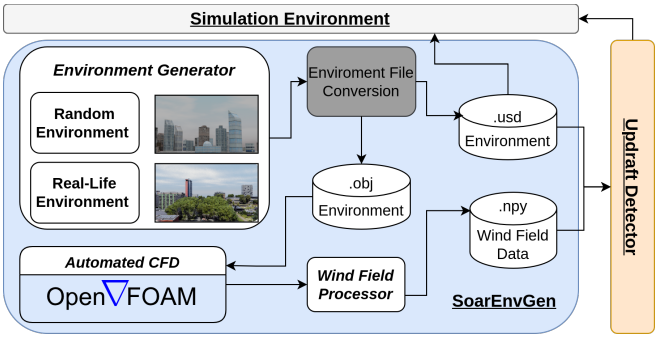


Fig. 7. The SoarEnv Module

1) 3D Environment Conversion and Simulation Setup: The SoarEnv pipeline provides two main methods for generating 3D urban environments, each serving different needs. The first method uses Blender's blosm add-on, which was shortly described in subsection III-B, to create realistic urban landscapes by importing geographic data, allowing for the precise modeling of real-world cities and environments [22]. This is particularly useful when simulating actual locations where environmental factors like wind interaction need to be studied in specific contexts.

The second method involves the generation of random urban environments using the custom Python random environment generator API within SoarEnv. This API automates the import of pre-modeled buildings, places them randomly within a defined field, and applies randomized textures to ensure visual variation. A key feature of the API is its ability to check for and prevent building overlap, which helps maintain a realistic urban layout. Additionally, the API generates a textured ground plane, completing the environment setup. Once the environment is created, it is exported in OBJ & USD format, ready for respectively CFD simulations in OpenFOAM and integration within the NVIDIA IsaacSim framework. Figure 8 shows a randomly generated environment, highlighting the variety in building textures and shapes.

These two methods together provide maximum flexibility in environment generation, whether for real-world applications or hypothetical scenarios. The random generation method, in particular, allows for the creation of vast amounts of diverse data, which is invaluable for training machine learning models. By simulating wind fields across varied environments, SoarEnv supports research into neural networks for updraft detection, as well as use cases for predicting wind behavior in complex urban settings and around buildings. This scalability is a needed and valuable feature for training neural networks.

2) CFD Simulations Using OpenFOAM: Once the 3D environment is generated, SoarEnv leverages OpenFOAM to conduct automated CFD simulations, capturing realistic wind field data. The simulation begins with setting up the mesh,

Fig. 8. Randomly Created Environment with SoarEnv package

where the 3D geometry from the environment file is used to define the simulation volume. Key parameters, such as the domain size, are configured based on the dimensions of the imported model, ensuring sufficient margins around the buildings to prevent boundary interference. This is achieved by dynamically adjusting the mesh and grid cell sizes according to the environment's geometry.

The wind velocity and direction are specified as part of the simulation settings. The OpenFOAM simpleFOAM solver, combined with the RANS turbulence model, is employed to handle the steady-state airflow simulations [26]. To create a comprehensive dataset, the simulation runs for a range of wind velocities and directions, generating a full map of possible wind conditions across the environment. All necessary configurations, including mesh generation and solver parameters, are stored and applied through automated scripts, streamlining the entire CFD workflow. During the mesh preparation phase, a refinement box is defined around key areas to ensure higher accuracy in wind field calculations. Key areas are regions where the flow field is expected to change significantly, such as around buildings and other geometric objects.

After setting up the simulation, SoarEnv uses its custom API to fully automate the OpenFOAM process, from mesh generation to solver execution. The API ensures seamless integration of all steps, applying the configured settings and managing the simulation workflow without manual intervention. The output is a detailed wind field, represented as vector data, which is saved for post-processing and visualization. These wind fields can then be integrated into NVIDIA IsaacSim or other platforms, enabling real-time interaction with precomputed airflow data.

E. Post-Processing and Wind Field Integration

Once the simpleFOAM solver completes the steady-state simulations, it returns a three-dimensional wind field, represented as vector data at each mesh point. These wind fields are then processed and integrated directly into the Osprey Simulator. The wind direction in the simulations is defined using cardinal points—North (N), East (E), South (S), West (W), and their intermediate directions NE, SE, SW, and NW—which are

mapped to specific angles relative to the positive x-axis in the global reference frame.

These angles are assigned as follows: North corresponds to $0°$, East to $90°$, South to $180°$, West to $-90°$, Northeast to $45°$, Southeast to $135°$, Southwest to $-135°$, and Northwest to $-45°$. By converting these angles to radians, the wind velocity vectors are calculated using trigonometric functions:

$$\begin{aligned} u &= -V\cos(\theta)\cos(\phi) \\ v &= -V\sin(\theta)\cos(\phi) \\ w &= -V\cos(\phi) \end{aligned} \tag{7}$$

where V is the wind speed, θ is the wind direction angle in radians, ϕ is the elevation angle, and u, v, and w are the wind velocity components along the x (east-west), y (north-south), and z (vertical) axes, respectively. The negative signs ensure that the wind vector points in the correct direction relative to the coordinate system. For example, a wind blowing from the north (N) with a speed of V would have components $(-V, , 0, , 0)$, indicating a wind moving towards the negative x-direction (westward).

This mapping aligns the wind directions with the global reference frame used in the simulation, where the x-axis represents the east-west direction (positive towards the east), the y-axis represents the north-south direction (positive towards the north), and the z-axis represents the vertical direction (positive upwards). By accurately defining the wind directions and their corresponding velocity vectors, the simulation captures realistic wind interactions within urban environments

In Figure 9 , Figure 10 and Figure 11, the wind field's updraft regions and flow field are clearly visible, showcasing the pipeline's ability to capture realistic airflow behavior around buildings. OpenFOAM, a well-known and reliable CFD tool, was used to generate these wind fields, and its established accuracy reduces the need for extensive verification in this context. To assess the qualitative accuracy of SoarEnv, the wind field was initially compared against flow patterns around square-shaped buildings, which are commonly studied in existing research [4, 3, 12]. These comparisons demonstrated that the resulting flow patterns closely resembled computational and experimental findings reported in the literature, providing confidence in SoarEnv's accuracy. While most urban buildings tend to be rectangular, it is assumed that the wind fields generated by SoarEnv will also be reliable for other building shapes. However, further quantitative validation may be required for more complex or unusual geometries.

To conclude, SoarEnv plays a critical role in the Osprey Simulator by generating realistic wind fields. With SoarEnv providing accurate airflow simulations, Osprey Simulator can be used to model orographic updraft recognition and optimize energy-efficient path planning.

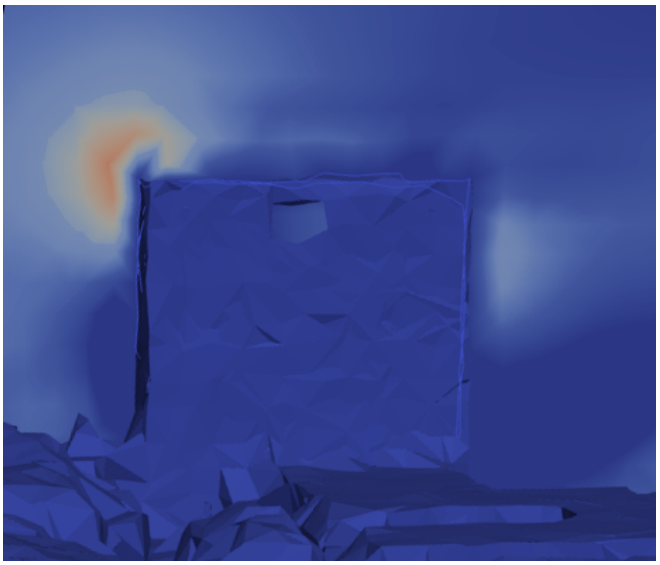


Fig. 9. TU Delft Aerospace Building corner slice vertical (Uz) wind vector

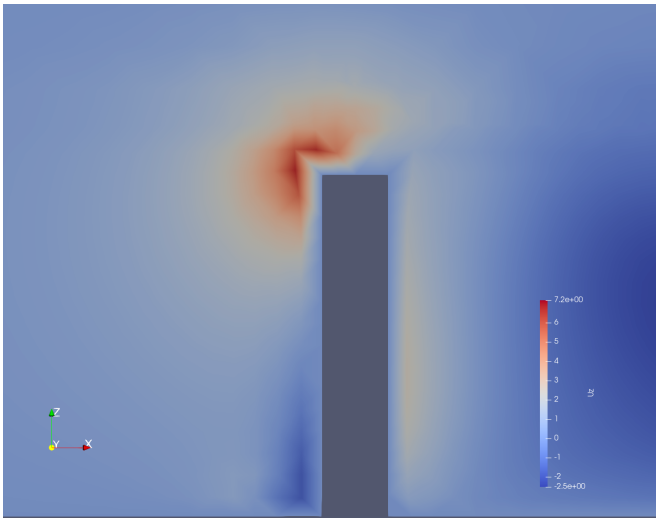


Fig. 10. Random Generated Building vertical wind vector

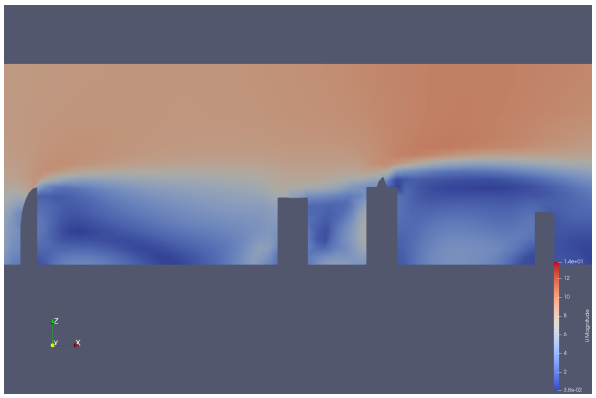


Fig. 11. Wind field shape

IV. Self-Supervised Learning for Updraft Detection

In the final part of the Osprey Simulator setup, we build a neural network capable of detecting soaring hotspots, where updrafts are likely to occur. These soaring hotspots are essential for energy-efficient path planning or zero-thrust hovering of fixed-wing drones. The focus is put on orographic updrafts, with the current database consisting mainly of urban environments. By using urban environments, the predictability of wind patterns around buildings and urban structures can be leveraged. The network will be trained using synthetic data generated within the simulator, specifically depth images that mark regions with vertical wind vectors. Depth images provide a pixel-wise representation of the distance from the drone to surrounding objects, which is critical for identifying terrain features, building structures, and other elements that influence wind dynamics. This section outlines the process of generating this data and preparing the network for updraft detection.

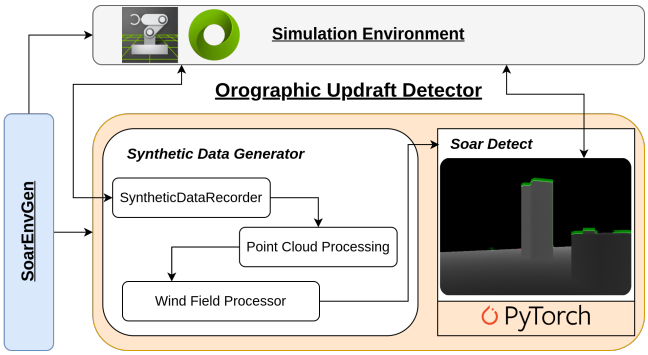


Fig. 12. The orographic updraft detection module

A. Data collection

To train a neural network effectively, a large and diverse dataset is required. The dataset will include the input depth images, wind direction, and labeled updraft regions. These updraft regions are represented by the wind velocity in the z-direction. Updrafts are specific to the sink rate of the drone; the sink rate of the drone needs to be lower than the vertical wind vector. Once this is the case, it can be considered an updraft region. Because the model predicts the vertical wind velocity at each pixel in the environment, it can be adapted for use with any fixed-wing drone, regardless of its specific C_l, C_d curve or sink rate. By setting a threshold based on the drone's sink rate, an updraft region can be identified as any location where the predicted upward velocity is equal to or greater than the sink rate. This flexibility makes the model broadly applicable across different drone configurations. The synthetic data generation pipeline will be leveraged to do this, as can be seen in Figure 12. The pipeline will, *de facto*, ensure that there is an automatic label generation process, making data collection exponentially scalable with the number and variability of environments.

The pipeline first sets up the environments—both random and real-life environments that have been created—and then randomly takes images within these environments while randomizing the camera locations and rotations. These rotations and positions are critical for the wind direction relative to each camera position; the wind direction will be one of the inputs to the neural network, as it has an important impact on predicting whether the building shows an updraft or downdraft (positive or negative vertical wind vector). Updrafts occur above buildings on the sides where wind hits the structure (see Section II, Figure 9 and Figure 10).

The synthetic DataRecorder class records the depth images, which represent the distance from the camera to objects in the environments. The choice of using depth images mainly comes from the ease of having only one channel, which lowers the computational effort for training, while still providing critical geometric information about the scene within the camera frame.

Thanks to the Isaac Sim framework, a point cloud can be generated for each depth image, eliminating the need for camera transformations. The PointCloudProcessor class reshapes the point cloud data into coordinates per pixel.

These coordinates are then used in the WindFieldProcessor, where the wind field data is processed for the coordinates per pixel. To make this computationally more efficient, this process has been pre-filtered by leveraging our knowledge of the behavior of orographic wind field data. The network will be interested in updraft regions above buildings, which allows us to pre-filter and label as follows. First, the building edges—mainly the rooftops—are targeted using OpenCV edge detection. Next, for each detected edge pixel's coordinate, an upward movement vector is applied. As shown in Figures 10 and 9, the optimal updraft generally occurs above the building edges. After applying the movement vector, the new coordinate is used to obtain an interpolated average value of the vertical wind vector from the wind field data. This value is then assigned to a range of pixels, which can be adjusted as needed, creating an updraft or downdraft zone. The resulting image has the same shape as the depth image, (H, W), but now the pixels contain the vertical wind scalar values. These "vertical wind images" or arrays will be used as labels during the neural network training process. This process is also visualized in Figure 13.

The input wind direction relative to the drone/camera body frame, the depth image, and the labels per image are now created, saved, and processed to be used for training.

B. Neural Network: SoarDetect

To effectively detect updraft regions from depth images in urban environments, a neural network that can capture both local spatial details and global contextual information is essential. For the network architecture design it was decided to utilize a modified U-Net architecture for this task due to its proven success in image segmentation and pixel-wise prediction problems [27]. The U-Net's encoder-decoder structure with skip connections allows it to learn hierarchical features and preserve spatial resolution, which is critical for identifying updraft regions that often occur near edges and rooftops. The network was implemented using Pytorch [28].

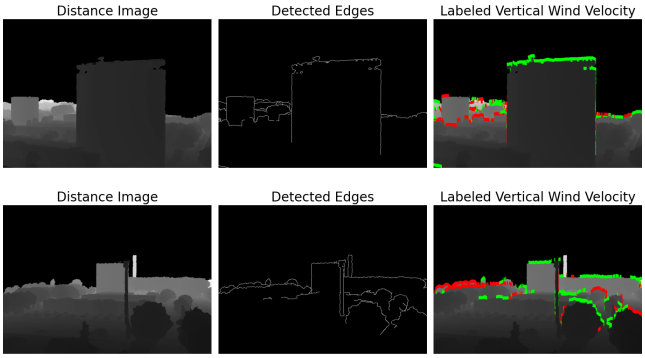


Fig. 13. Synthetic Data Vertical Wind Image Process

1) U-Net Modification and Structure: The input to the network consists of a single channel representing the depth image, while the wind vector components (V_x, V_y) are incorporated at the bottleneck of the U-Net architecture. By introducing the wind vector components at the deepest layer of the network the model gains explicit information about global wind conditions while maintaining focus on spatially localized interactions between wind and environmental structures. This approach allows the network to learn how large-scale wind patterns influence vertical wind velocities, particularly around complex geometries such as building edges, where updrafts are most pronounced.

The modified U-Net architecture comprises an encoder and a decoder connected through skip connections at each level. The encoder path consists of four blocks, each containing two convolutional layers with batch normalization and ReLU activation, followed by a max-pooling layer for downsampling. The number of feature channels doubles at each downsampling step, starting from 64 and reaching 512 in the bottleneck, enabling the network to capture features at multiple scales essential for understanding both fine details and broader context.

At the bottleneck, the wind vector components are transformed through a fully connected layer to produce a feature representation, which is then spatially broadcasted and concatenated with the encoder's output. This integration enriches the bottleneck feature map with wind-specific global context, facilitating the network's ability to learn complex aerodynamic interactions.

The decoder path mirrors the encoder but replaces max-pooling with upsampling layers. Employing bilinear interpolation for upsampling due to its simplicity and ability to reduce checkerboard artifacts commonly associated with transposed convolutions [29]. After each upsampling step, the feature map is concatenated with the corresponding feature map from the encoder via skip connections, enabling the network to reconstruct spatial details necessary for precise localization of updraft regions. The final layer is a 1×1 convolution that produces a single-channel output representing the predicted vertical wind velocities (V_z). By integrating wind vector components at the bottleneck, the network effectively combines global wind conditions with spatially detailed environmental features, enhancing its ability to detect soaring hotspots for energy-efficient drone flight planning.

C. Input and Output of the Model

The modified U-Net model accepts a single input tensor consisting of three channels:

1) **Depth Image**: A single-channel image representing the distance from the camera to objects in the scene. The depth images are normalized to ensure numerical stability and consistent scaling across the dataset.
2) **Wind Vector**: A two-dimensional vector $\mathbf{v} = [v_x, v_y]$ representing the wind velocity components relative to the drone's frame of reference. The wind vector is incorporated into the network at the bottleneck, where it is transformed into a feature representation using a fully connected layer. This feature representation is expanded spatially to match the bottleneck dimensions and concatenated with the encoder's output. By providing the wind vector at this stage, the network can leverage both global wind conditions and spatially detailed environmental features to predict updraft regions accurately.

The depth images are normalized to the range $[0, 1]$ to ensure training stability [30]. These normalized depth images are combined with the wind vector features at the bottleneck to form the network's complete input. The vertical wind images serve as the ground truth outputs, providing per-pixel labels for training, as discussed in subsection IV-A.

The output of the model is a single-channel image with the same spatial dimensions as the input depth image and the ground truth labels. Each pixel value in the output represents the predicted vertical (z-direction) wind velocity at its corresponding spatial location. Pixels with positive values indicate updrafts, while negative values denote downdrafts. The model's predictions closely resemble the synthetically labeled data, enabling precise identification of soaring hotspots and downdraft regions for energy-efficient drone flight planning.

D. Training

The goal of the training process is to enable the model to learn how geometric features captured in depth images and global wind conditions represented by wind vectors combine to influence vertical wind velocities, particularly in identifying localized updraft and downdraft regions caused by aerodynamic interactions with complex environmental structures. The model was trained on a computer with an Intel Core i7 10th Gen CPU and an NVIDIA GeForce RTX 2080 Super with Max-Q Design, featuring 8 GB of memory.

The synthetic dataset used for training was generated from two distinct environments: a randomly generated environment and a real-world environment based on the TU Delft campus. Both environments were simulated under varying wind velocities and drone positions. he randomly generated environment consisted of procedurally created building shapes, including cubes, spheres, rounded roofs, triangular roofs, and domes, offering a diverse range of aerodynamic challenges. In contrast,

the TU Delft campus provided realistic geometric details with well-defined architectural structures. While in contrast to that the TU Delft campus provided realistic geometric details with well-defined, complex architectural features. This combination ensured high variability in aerodynamic conditions, enriching the dataset and allowing the model to generalize effectively. As a result, the network achieved low training loss and was able to predict vertical (z-direction) wind velocities with high accuracy, even in regions with intricate flow dynamics, such as around building edges.

1) Loss Function and Optimization: For the loss function the Mean Squared Error (MSE) loss function was implemented to measure the difference between the predicted and actual vertical wind velocities. The MSE loss is suitable for regression tasks where the goal is to predict continuous values [30]. Next, for the optimizer it was decided to make use of the Adam optimizer. The decision was made because of its ability to handle sparse gradients and adapt the learning rate during training, which enhances the convergence speed and stability of the training [31]. The training loss can be seen in Figure 14

2) Hyperparameters: The model was trained using a learning rate of 1×10^{-4}, which balances the need for steady convergence with the avoidance of overshooting minima. Due to computational constraints, a batch size of 2 was used; although small, this batch size was sufficient for the model to learn from the data.

The network was trained for 25 epochs using a dataset of 5,800 samples. The random environment contributed 4,000 samples, derived from 400 images simulated across 10 different wind velocities. The TU Delft campus environment provided 1,800 samples, generated from 600 images simulated at three different wind velocities.

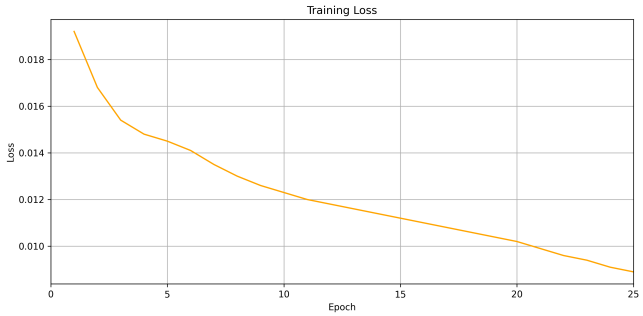


Fig. 14. Training Loss SoarDetect

E. Testing and Results

After training, the model's performance was evaluated on a separate test dataset consisting of depth images and wind vectors that were not used during training.

The model demonstrates generalization and correctly predicts updraft regions. However, there are cases where it incorrectly identifies certain features as edges, such as the border of the ground plane being falsely recognized as a rooftop line. Increasing the diversity of the dataset by introducing more

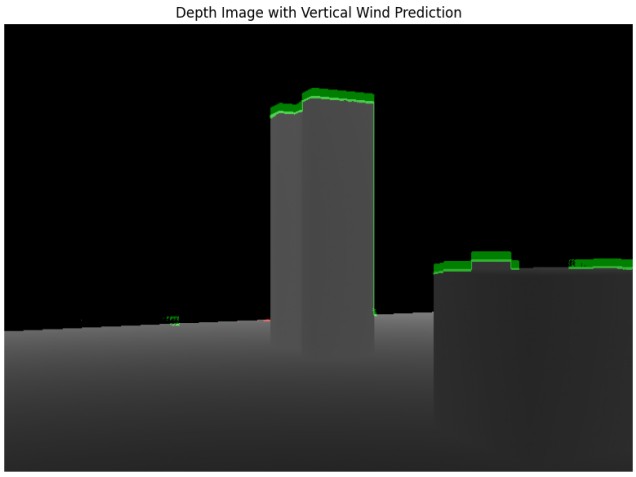


Fig. 15. Updraft Prediction

environments is expected to mitigate such issues. The model also performs better with the randomly generated environment compared to the TU Delft campus environment, likely due to the simpler geometries in the former. The TU Delft environment presents more complex shapes, which challenge the model's predictive capabilities.

One notable observation is the model's ability to adapt its predictions to varying wind velocities. For instance, at low wind speeds of 1 m/s, where updraft regions are minimal, the model accurately predicts this lack of significant updrafts. Conversely, at higher wind velocities, it effectively identifies larger updraft regions. Additionally, the model recognizes downdraft regions, a critical capability for realistic predictions. This is evident in scenarios where the drone faces a building front with a headwind, correctly predicting the absence of an updraft on that side. For flight planning, this concept can be extended by implementing an algorithm that accounts for the likelihood of an updraft on the opposite side of the building. This is particularly useful when the drone is positioned below the roof surface and cannot directly observe the other side. An example of a predicted downdraft is shown in Figure 16.

It is important to note that the test data still originates from the same two environments used during training. To achieve more comprehensive results, evaluations on entirely unseen environments would be necessary.

F. Future Work

Future work involves validating the model with real-world data to assess its performance outside simulated environments. This validation will help assess the model's generalizability and practical utility in actual drone operations, ensuring that it can accurately detect updraft regions under varying real-world conditions.

Testing the model on entirely new environments beyond those used during training is also essential to further evaluate its ability to generalize to unseen geometries and wind conditions. Expanding the dataset with a broader range of building

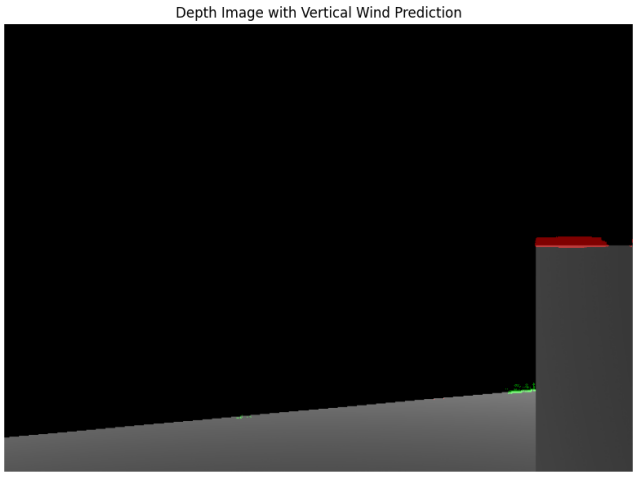


Fig. 16. Downdraft Prediction

shapes and wind scenarios would address current challenges, such as false edge detections, and enhance robustness.

Additionally, integrating the model into real-time drone systems could enable practical deployment for energy-efficient flight planning. Algorithms could be developed to infer updraft regions in unseen areas, such as predicting updrafts on the opposite side of a building when only one side is visible.

Lastly, due to computational constraints, the environments simulated during training were limited in size and complexity. Access to more powerful hardware would allow for generating larger and more detailed environments, further improving the model's ability to learn and generalize.

V. Conclusion

Osprey Simulator is a platform designed to advance research in fixed-wing drone operations by simulating real-world environments with the goal of enabling learning for autonomous soaring. The simulator includes a detailed fixed-wing aircraft model, incorporating its own control modes and physics to accurately replicate the dynamics of real-world flight.

Central to this is SoarEnv, which generates accurate, detailed wind fields, enabling critical applications such as orographic updraft recognition and energy-efficient path planning. Utilizing Osprey Simulator and SoarEnv, synthetic data was collected by combining the generated wind fields with depth images and the simulator's ability to get the wold coordinates of the objects in the image. Providing a rich dataset of depth images and corresponding wind vectors. This data served as the foundation for training the modified U-Net neural network, SoarDetect, designed to predict vertical wind velocities from depth images and wind information.

By integrating these realistic environmental factors and advanced neural network models, Osprey Simulator provides a unique opportunity for researchers to test and optimize fixed-wing drone performance in complex, dynamic conditions. The successful training of SoarDetect demonstrates the platform's capability to facilitate significant advancements in autonomous drone technology, particularly in enhancing energy efficiency through updraft exploitation.

References

[1] Michael Allen. "Autonomous soaring for improved endurance of a small uninhabitated air vehicle". In: *43rd AIAA Aerospace Sciences Meeting and Exhibit*. 2005, p. 1025.

[2] Abdulghani Mohamed et al. "Opportunistic soaring by birds suggests new opportunities for atmospheric energy harvesting by flying robots". In: *Journal of the Royal Society, Interface* 19 (Nov. 2022), p. 20220671. DOI: 10.1098/rsif.2022.0671.

[3] Caleb White et al. "A feasibility study of micro air vehicles soaring tall buildings". In: *Journal of Wind Engineering and Industrial Aerodynamics* 103 (2012), pp. 41–49.

[4] Simon Watkins et al. "Towards autonomous MAV soaring in cities: CFD simulation, EFD measurement and flight trials". In: *International Journal of Micro Air Vehicles* 7.4 (2015), pp. 441–448.

[5] Tom Suys et al. "Autonomous Control for Orographic Soaring of Fixed-Wing UAVs". In: *arXiv preprint arXiv:2305.13891* (2023).

[6] Sunyou Hwang, Bart DW Remes, and Guido CHE de Croon. "AOSoar: Autonomous Orographic Soaring of a Micro Air Vehicle". In: *arXiv preprint arXiv:2308.00565* (2023).

[7] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021. arXiv: 2108.10470.

[8] Marcelo Jacinto et al. *Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation*. 2023. arXiv: 2307.05263 [cs.RO].

[9] Caleb White et al. "The soaring potential of a micro air vehicle in an urban environment". In: *International Journal of Micro Air Vehicles* 4.1 (2012), pp. 1–13.

[10] Alex Fisher et al. "Micro air vehicle soaring in urban environments". In: *2016 Australian Control Conference (AuCC)*. IEEE. 2016, pp. 9–14. DOI: 10.1109/AUCC.2016.7867924.

[11] Abdulghani Mohamed et al. "Scale-resolving simulation to predict the updraught regions over buildings for MAV orographic lift soaring". In: *Journal of Wind Engineering and Industrial Aerodynamics* 140 (2015), pp. 34–48.

[12] Joshua C Nathanael, Chung Hung J Wang, and Kin Huat Low. "Simulation of Wind Field in a Building Complex for Evaluation of the Wind Effect Along UAS Flight Path". In: *AIAA AVIATION 2023 Forum*. 2023, p. 4096.

[13] Jake Tallman. "Soarnet, Deep Learning Thermal Detection for Free Flight". PhD thesis. California Polytechnic State University, 2021.

[14] Chris PL de Jong et al. "Never landing drone: Autonomous soaring of a unmanned aerial vehicle in front of a moving obstacle". In: *International Journal of Micro Air Vehicles* 13 (2021), pp. 1–12.

[15] Shital Shah et al. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *Field and Service Robotics*. 2017. eprint: arXiv:1705.05065. URL: https://arxiv.org/abs/1705.05065.

[16] Yunlong Song et al. “Flightmare: A flexible quadrotor simulator”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 1147–1157.

[17] Brian Karis. “Real Shading in Unreal Engine 4”. In: *Proceedings of the ACM SIGGRAPH 2013 Courses*. ACM, 2013.

[18] Nathan Koenig and Andrew Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2004, pp. 2149–2154.

[19] Unity Technologies. *Unity 5 Manual*. Available at: https://docs.unity3d.com/Manual/. 2015.

[20] Hajo Erwich. “GSL-Bench: High Fidelity Gas Source Localization Benchmarking”. In: (2023).

[21] Nicholas Kakavitsas et al. “Quadrotor Flight Simulation in a CFD-generated Urban Wind Field”. In: *2024 IEEE Aerospace Conference*. IEEE. 2024, pp. 1–8.

[22] vvoovv. *Documentation for Blosm addon*. https://github.com/vvoovv/blosm/wiki/Documentation. Accessed: 2024-05-19. 2024.

[23] Blender. *Blender - Simulations*. Accessed: 26 August 2023. URL: %5Curl%7Bhttps://www.blender.org/features/simulation/#fluids%7D.

[24] Ger J.J. Ruijgrok. *Elements of Airplane Performance*. ISBN: [Insert ISBN Here, if known]. Delft University Press, 2009.

[25] MIT Unified Thermodynamics Course. *Airspeed and Propeller Advance Ratio*. Accessed: [Insert Date Here]. n.d. URL: https://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html.

[26] C. Peralta et al. “Validation of the simpleFoam (RANS) solver for the atmospheric boundary layer in complex terrain”. In: *Wind Energy Science* 3 (2013), pp. 237–247.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 234–241.

[28] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in neural information processing systems* 32 (2019).

[29] Augustus Odena, Vincent Dumoulin, and Chris Olah. “Deconvolution and Checkerboard Artifacts”. In: *Distill* (2016).

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[31] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (2015).

Part II

Literature Study

Summary

The technology evolution of Unmanned Aerial Vehicles (UAV's) has undergone significant advancements, with rapid progression showing no signs of slowing down. Different industries, researchers and even governments are looking to use UAV's for all kind of purposes, where most of them are limited by their endurance. Due to batteries already having excellent performance and efficiency other ways have to be found to increase their endurance. One solution that is currently being researched and is a big part of this literature study is the bio-inspired way of using bird-soaring techniques to harvest atmospheric energy. This report will focus on orographic soaring which is a specific soaring technique that's possible when wind fields are created by sufficiently large (human created) structures.

Although some research has been performed on thermal soaring for UAV's, the research on orographic soaring has only been gaining momentum over the last couple of years. This report tries to prepare the author and reader with enough knowledge for the upcoming thesis project which will focus on orographic updraft recognition within a self-created simulated environment.

The literature study starts off by giving a current update on soaring techniques and delves further into the research that has been performed on orographic soaring. It shows that to the best of the author's knowledge the first significant contribution was in 2012 where a research team recognised the possibilities of orographic soaring. The emphasis in the orographic soaring research was more on exploring possibilities as well as some tests which conducted. An autonomous soaring algorithm was created to let a drone position itself within the wind field, however nothing was done regarding updraft recognition.

The chapter closes of with a description of how orographic wind fields behave. It can be concluded that, given the wind conditions and object shapes, these fields always take similar form which could be beneficial for recognition purposes.

The other main part of the report, is the chapter that focuses on the machine learning of the thesis project. Here it is seen that a clear gap exists in the field of updraft localization. A review has been done of the current state of the art techniques, while similar application papers were discussed. Historically Machine Learning (ML) models have leaned heavily on supervised learning, requiring significant amounts of labeled data. On the other hand, self-supervised learning, a more novel approach, presents an alternative for this. By using the data's inherent structure, it loses the need for a time consuming annotation process. It should be noted that here the challenge lies in the novelty of the use case as well as getting the same accuracy as for supervised models. Bridging the gap between the previously mentioned techniques is Semi-Supervised learning. Semi-Supervised learning could present as a compelling option where it could harness the best of both.

As all different techniques are explored, a decision will have to be made between the tried-and-tested annotating intensive supervised learning or the newer, more intricate self-supervised learning which shows a lot of potential.

The report closes off with an in-depth examination for a simulator pipeline. Within this chapter, the efficacy of the Pegasus Simulator is shown. Although some fine-tuning will be needed, once tailored for the fixed wing updraft recognition task, the Pegasus Simulator and the underlying NVIDIA's Isaac Sim application are a good foundation to build on during the thesis project. Next to a fixed wing integration, the simulator will need an integration for a Computational Fluid Dynamics (CFD) pipeline. Additionally, configuring the chosen ML technique with the simulator also remain something that has to be addressed.

The literature study also gave the research question for the upcoming thesis project which is as follows:

How to autonomously pick & recognize orographic soaring locations from a distance for an UAV by using extrareceptive inputs and how can we learn this in a simulated (urban) environment?

While some aspects of this question have been addressed, the comprehensive answering and practical application will be explored in the subsequent thesis project.

1

Introduction

The past few years some research has focused on using soaring techniques inspired by birds for UAV's. This research has been performed with the goal of solving the long-endurance challenge for battery-powered aerial vehicles. Throughout this literature study it will become clear that multiple autonomous control algorithms have been created for the soaring of UAV's. In addition to these created control algorithms, autonomous cross country challenges have been performed to test the advantage of soaring for optimal path planning. Another focus of soaring techniques is hovering which is done by making use of orographic lift.

Both these ideas have been performed in simulation as well as experimentally. This literature study will now prepare the author for a not well researched section of autonomous soaring, it will provide the basics to start the thesis project that focuses on creating a simulation for orographic updraft localization in an urban environment.

The idea of soaring for UAV's was first introduced in 2005 by Allen, here Allen recognised to possibilities for UAV's by increasing their endurance between 2 and 14 hours [4]. By now the research on soaring has been explored much further with the focus being on the techniques of dynamic and static soaring for applications in hovering and optimal path planning. To start off the literature study, in the following sections the research question will be formed and the focus points for the thesis project will be set.

1.1. Research Question

For the continuation of the thesis research following this literature study report, it is of high importance to formulate a research question. The research question will be addressed at the end of the project. The question is as follows:

How to autonomously pick & recognize orographic soaring locations from a distance for an Unmanned Aerial Vehicle (UAV) by using extrareceptive inputs and how can we learn this in a simulated (urban) environment?

Now, the research question contains some key words which will on their own summarize the research that is done in this report. Breaking down these key words and giving their definition is important as this way there is a clear understanding of what is meant, which minimizes any ambiguity. These key words will on their own break down in further sub-questions that will help further with the ambiguity of the research question.

- Orographic Soaring: This report will give an introduction to the different soaring techniques in existence today. Furthermore it will show the distinction between static soaring and dynamic soaring, where orographic soaring is a subcategory of static soaring. **Sub-Questions**
  - How will the orographic lift be simulated?
    - How does CFD come into play here?

        * Is CFD compatible with the chosen simulator?
        * How will the environment be created?
- UAV: The fixed-wing Unmanned Aerial Vehicle that will act as agent in the simulation. Fixed Wing, with a high aspect ratio are a necessity for soaring vehicles.

  Sub-Questions

  - How will the UAV be implemented into the simulator?
    * Which simulator framework will be easiest to implement the fixed wing?
- Autonomously: The detection of good soaring places will be done autonomously. This will result in looking into different Machine Learning techniques to achieve this. The thesis project will focus on the autonomous detection of soaring locations and will not develop a soaring controller as this is out of scope for the thesis project.

  Sub-Questions

  - What techniques will be needed for this?
    * From all the computer vision algorithms which one will work best?
    * Has this been done before for other soaring techniques than updrafts?
    * If it has not been done before, how will it be implemented for this project?
  - What are good soaring locations?
    * Will the focus be put on buildings alone?
    * Are there other orographic soaring locations in urban cities besides buildings?
- Extrareceptive Inputs: It is important to have and select realistic sensors that will be able to use the Extrareceptive inputs from the environment in a way the autonomous detection works. **Sub-Questions**

  - What will these inputs be, which sensors will be needed?
    * What input sensors are realistic to use in real-life as well as input sensors that are available in the simulator environment of choice?
    * How many sensors will be needed, knowing that having too much will increase the computational load of the UAV and the ML algorithm?
- Simulated Environment: The project will not focus on real-life tests but on the creation of a simulation environment. This environment should be realistic enough that in the end it can be used to train in the simulator and then can be deployed with an accurate model for real life applications. At first, the simulated environment will be an urban environment.

  Sub-Questions

  - Should there be other environments than the urban?
    * Once the urban is created, and there is still time left, would it be beneficial to create a coastal/sea environment to simulate the work in [28]?
  - What will the best simulator be for this case?
    * The research shall have to focus on game rendering engines, which one will come out on top?
    * What kind of simulators have been used in other applications in the MAVLab research team?

These key-words also have a further usage, which helping to focus during the research, and setting said research objectives. These research objectives are explored in section 1.2.

1.2. Focus during research

The research is done for the Never Landing Drone division within the MAVLab. The research is done in preparation for the Thesis project that will follow, it is therefore important to set some objectives during the research study. These objectives will also be similar to the outline of the report, setting well-defined objectives will accomplish a good start of the thesis project. The setting of certain research objectives will be derived from the research question and the key-words explained in the last section.

Key literature study focus areas:

- Understanding the control of orographic soaring
  - Look into the fixed-wing configuration for the orographic soaring
- Achieve an understanding of updrafts inside simulator frameworks. Has any research been done? If yes get up to date with said research.
- Have a good understanding of the different simulator frameworks and environments currently available for the application within this project.
- Look into the different Deep Learning technologies that have been used for updraft recognition. If none has been done, look into techniques that seem promising.

Completing the above tasks will result in a good preparation for the thesis, it is however also good to set the main goals that will need to be completed during the thesis. Focusing on these goals will help with the planning and the structure of the thesis as is discussed in section 5.2

- Create a pipeline to introduce wind fields into a simulated environment
- Create an agent within environment that has a soaring controller included as well as extraceptive sensors that can be used for detection
- Look into and create a ML algorithm, then train the agent in recognizing good locations for soaring

2

Soaring Techniques

This chapter will give an overview of the research that has been done on the different kind of soaring techniques currently being used. For soaring there are two different distinctions that have to be made and that's static soaring and dynamic soaring. section 2.2 explains how static soaring is defined as the gaining of atmospheric energy due to a vertical wind component which is called an updraft. Static soaring also has two sub categories, thermal soaring which is widely used by birds and gliders, and orographic soaring which is also used by birds in all different kind of forms [40]. Orographic soaring is what will be the main focus for the thesis project.

An overview of dynamic soaring will be given in section 2.4. This section will give a small overview and will only give the main pointers as it has been extensively researched and is not the focus within the thesis project.

This chapter will also give a clear overview of theoretical framework of the subject and the different concepts that need to be understood for the thesis. The chapter will end with a summary of the key findings and from this chapter the common themes as well as the gaps in literature will be shown in section 5.1.

2.1. Atmospheric energy Harvesting

The need for autonomous soaring comes from the endurance problem for UAV's where the current state of the art batteries are already very efficient. This means that extra endurance will have to be created with something other than increasing battery efficiency, a deeper look into atmospheric energy harvesting.

Already in 2005, before the revolution in drone and battery technologies, M. J. Allen recognized the need for improved endurance of small UAV's [4]. It was recognised that soaring techniques commonly used by birds & glider pilots to improve their range, endurance, energy savings and cross-country speed, can also be used by UAV's. Using these techniques is also known as atmospheric harvesting, as the energy is harvested by making use of atmospheric conditions.

Atmospheric energy harvesting will only be possible if one of the two following conditions listed below are met. Where there is most definitely a high chance of both conditions occurring at the same time [40].

1. The wind field components are in an upward direction
2. The wind field varies spatially or temporally

These two conditions can also be transformed into 2.1 which shows the quantification of the distinct opportunities for atmospheric energy harvesting. The equation shows a clear distinction between the two different types of soaring. Both will be explained into further detail in respectively section 2.2 and section 2.4.

$$\frac{\mathrm{d}e}{\mathrm{d}t} = \overbrace{\frac{(T-D)V}{m}}^{\text{drag losses of net thrust}} + \overbrace{gW_U(s,t)}^{\text{Static soaring}} - \overbrace{\left[\boldsymbol{V}\cdot\frac{\partial \boldsymbol{W}}{\partial t} + \boldsymbol{V}\cdot\frac{\partial \boldsymbol{W}}{\partial s}\frac{\mathrm{d}s}{\mathrm{d}t}\right]}^{\text{dynamic soaring}}, \tag{2.1}$$

2.1.1. UAV configuration

While enjoying the opportunities offered by atmospheric energy harvesting would be important for any kind of autonomous aerial vehicle, the UAV's needs to satisfy certain conditions. Because of this, the result is that the UAV's dynamics should be fixed wing to receive the most energy for these techniques.

The specifications of two examples of fixed wing drones, that have successfully used soaring techniques can be seen in Table 2.1. Both of these were successfully used in research that will be discussed later on in subsection 2.2.2.

Table 2.1: Fixed-wing drone specifications for soaring [25], [28]

	Properties Parrot Disco UAV	Eclipson model C
Wingspan [mm]	1150	1100
Mass [g]	750	716
Aspect ratio [-]	/	6.9

The table shows that these drones have a low weight, which correlates to a lower sink-rate. Furthermore having slender wings, thus having an higher aspect ratio is also more beneficial for soaring. In Figure 2.1 a top view can be seen of the RigiTech drone which is a Vertical Takeoff and Landing (VTOL) fixed-wing drone that has been designed with as goal to have long range package delivery. Although the specs differ a lot with the ones seen in Table 2.1, keeping in mind that after some efficiency improvements soaring techniques could also be used by such like drones for commercial applications.

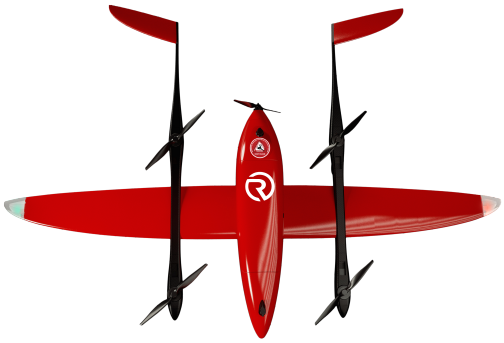


Figure 2.1: Top view of the Rigitech Eiger UAV [50]

Table 2.2: Specifications for the Rigitech Eiger UAV [50]

	Rigitech Eiger UAV
Wingspan [mm]	2240
Mass [g]	10,500
Aspect ratio [-]	/
Endurance 100 km, 1kg payload [minutes]	57 -70

It is also important to state that these opportunities do not limit themselves to urban environments and dunes, as research has shown that exploiting updrafts from thermals and orographic soaring has been used for monitoring solar parks, using ships, ... [40]. Further down the line this research can even be important for space missions as the recent NASA ingenuity helicopter mission success leaves open the need for extra endurance, range etc... [40].

2.2. Static Soaring

As mentioned in section 2.1 static soaring needs an upward vertical to have a positive specific energy flow, this also means that energy can be lost in a downdraft which has a downward wind component. The strategy of static soaring is particularly common among large birds, such as eagles and vultures [61]. Static soaring can be split up into two main components, Thermal Soaring and Orographic Soaring. While thermals are created by temperature differences in the air, the vertical wind component for

orographic soaring is generated by an object, creating orographic lift [40].

The strategies that are shown by soaring bird show a versatility that adapts between thermals and orographic soaring [61] [40]. This underscores the possibility that UAV's can employ similar techniques, capitalizing on similar opportunities presented by monitoring and detecting anthropogenic structures. This aligns directly with the primary aim of this research, as elaborated in chapter 1.

2.2.1. Thermal-soaring

Already introduced in the paragraphs above, thermal-soaring is currently the most widely used and researched atmospheric energy harvesting technique. Thermal-soaring is a type of soaring flight in which a bird, UAV's or glider pilot maintains or gains altitude by exploiting the so called thermals [3].

A thermal is an area of rising are that is caused by the difference in temperature between the warm air just above the ground and the "colder" air higher up. As the sun unevenly warms the Earth's surface, think of asphalt roads in the summer, parking lots, deserts or even solar parks as seen in [3][40]. The difference in the density of the air due to the temperature difference, the less dense, warmer air rises and creates a thermal, taking up soaring animals or UAV's with it.

Thermals are areas of rising air that are caused by the uneven heating of the ground by the sun. When a bird or flying robot flies into a thermal, the air beneath its wings is rising. This creates a difference in pressure between the air beneath the wings and the air above the wings. This pressure difference creates lift, which allows the bird or flying robot to maintain altitude without flapping its wings [67].

The research on thermal-soaring is extensive and, as said before, was first introduced by Allen in [4]. Allen recognised that the technique birds used was an untapped opportunity and the paper provided a systematic approach to the problem of autonomous soaring and demonstrated the potential of this technique for improving the endurance of UAV's [67].

Thermals are typically strongest in the morning and evening, when the sun is low in the sky. The size of a thermal can vary, but they can typically support a bird or flying robot weighing up to a few kilograms [3]. Also important to keep in mind is that, thermals can last for a few minutes or up to an hour, depending on the conditions.

As stated in the beginning of this chapter, Allen was the first to recognize the possibility of energy harvesting inspired by birds in [4]. The simulations of Allen, were all done with static soaring techniques and more specifically only thermals.

Allen followed up this research in [5] where outer loop soaring guidance and control controller was added to the autopilot. In this report the results of a real-life experimental test was performed, where the UAV's was able to climb 172m on average while exploring and exploiting thermal updrafts. This is now known as the first successful soaring guidance and control algorithm as well as the first successfully performed test. It should be known that for the autopilot, the updraft locations were known.

This research was followed up by different autonomous cross-country challenges which made use of thermals to gain height and therefore extend the duration of flight. Examples of this can be found in [43], [9], [68], [48]

Where in the first it is a cross-country soaring approach with hierarchical Reinforcement Learning (RL), the second path planning with heuristic search and the third RL for trajectory generation.

Again it should be noted that all three cases did not display any form of updraft detection and all thermal locations were known during flight. The controller was learned how to optimally use these thermals, not detect them.

In [60], a thermal detection neural network was created. However because of the unpredictable nature of thermals and its dependency on temperature and other weather conditions, the networks had to be trained on a dataset of different satellite images and their corresponding weather data. Although very interesting, the nature of orographic soaring locations and its predictability will make sure that such extensive data collection is not needed. More on orographic soaring as well as possible ML techniques in respectively subsection 2.2.2 and chapter 3.

As the research in thermals is not really relevant for the goal of the thesis project, this section will stay limited. However a deeper look into the possible RL techniques that were used for thermals in section 3.2, as this could prove useful for detection orographic soaring. It should also be remembered that like in the ”cross-country challenge” the research in thermals has mostly been done with the goal of optimal path planning in mind.

2.2.2. Orographic soaring

Orographic soaring is caused by horizontal wind vectors colliding with some kind of object and resulting in a vertical wind vector around the object, thus creating an updraft region around the structure. These objects can come in the forms of slopes, dunes, buildings, moving objects(ships) and mountains. Compared to thermal soaring which can give updrafts of hundreds of meters, orographic soaring only has updraft regions at lower altitudes around said objects [40]. However these objects give opportunities for energy harvesting in urban cities and coastal places, which are the environments that will be most used by UAV’s once applications as drone package delivery and drone building surveillance, surveillance become more prevalent [51]. The other advantage is that orographic soaring locations are more predictable to localize than thermals making it more advantageous for cross-country path planning [66].

In this section the current research on orographic soaring with UAV’s will be given. It will become clear that the focus of the autonomous soaring is mainly being applied to two applications, firstly in hovering. Orographic soaring for hovering means that when done correctly UAV’s can use 0% throttle and thus increasing their flight endurance this way .The other of the two main applications is using orographic soaring within path planning, here the idea is that a drone will choose the most energy efficient path by making use of orographic soaring locations. Recognizing these orographic soaring locations is where the follow-up thesis research will become important.

In [25], Hwang showed the wind tunnel experiment of slope soaring. Here it showed for the first time an autonomous soaring algorithm that positioned itself within an orographic updraft region without priori knowledge of the wind field. This report puts focus on two main aspects, the controller and the AOSearch algorithm. For the controller Hwang succeeded in creating a single controller for the navigation and soaring, this was done by adopting an Incremental Nonlinear Dynamic Inversion (INDI) controller with control allocation [25]. Control allocation refers to the process of distributing commands among multiple actuators in a system to achieve a specific control goal [27]. In this study, it is done by using Weighted Least Squares optimization to prioritize controlling pitch over thrust, aiming to minimize thrust for the outer loop controller [25].

This research is especially important for the follow up thesis project of this report, as once the updraft is localized, this algorithm could be suitable to have the UAV’s position itself correctly in the wind field.

UAV’s hovering is inspired by birds as well, Kestrels can be observed often to soar above dunes without using any ”throttle”, not flapping their wings [25][45]. Kestrels use this technique for hunting and are therefore hovering to surveil their prey, this analogy comes nice into place as now UAV’s can be used with orographic soaring for surveillance of different kinds as well [62].

Another key aspect of the literature study in preparation of the thesis project is the research on soaring in urban environments. The feasibility of this as well some simulations on this were performed by a team at the RMIT University in Australia, they first performed feasibility studies in [65] where a building representative for an urban environment was selected to simulations in [66]. In [65], C. White already recognized that due to performed tests the orographic lift will be between 15-50% of the wind speed at the soaring height, this is later confirmed in [64] & [41]. A detailed description of the orographic wind field of the report will be discussed in section 2.3.

The further research of Watkins introduced CFD analysis and how this influences the wind flow wield, the details of this will be discussed further in section 2.3. Important in this research is how, as to to the author’s current knowledge, this was one of the first detailed analysis’s for orographic soaring in Urban environments. It showed with simulations as well as experimental tests that orographic soaring is viable for optimal path planning [64]. Here the viability of using CFD proves its important role and it will play a big role in the upcoming thesis project.

In [19] the author sets outs the three main challenges for orographic soaring in urban environments for

the first time. The viewed challenges are the following [19]:

- The prediction and sensing of orographic updrafts as well as a strength estimation
- The optimal path planning in an environment while making use of updraft locations withing a mission
- Use the harvested energy from updrafts to increase in altitude or find another way of harvesting the energy.

The first recognized challenge is really what will be explored further on in the follow-up thesis project. Furthermore the paper tries to answer these challenges by giving a report of the author's attempt in solving them.

The occurrence of orographic updrafts is much more predictable than it is for thermals, giving it an advantage in localizing it [19]. As orographic soaring is understandable and occurs at structure under certain wind conditions, helping an agent understand this will be important and the main goal [19]. The technique that's investigated here is to first perform numerical calculations on with CFD on the environment and use these predictions for the autonomous soaring, this is however not really efficient once another environment is used [19]. In chapter 3 research will be discussed on how an agent can learn from these simulated environments and can use it in environments on which it was not trained. Doing this will be one of the main task for the thesis project and will answer the research question in section 1.1.

[41] shows the investigation of A.Mohamed et al. where the authors try to investigate the use of CFD for understanding the updraft intensities around buildings and in complex environments. Besides this they try to achieve a general understanding of the wind flow fields in these complex environments. This paper will be explained into more detail in section 2.3.

In 2021 The MAVLab Team at the TU Delft succeeded in achieving hovering flight in front of a moving ship [28]. [28] reported to achieve autonomous soaring at 0% throttle in the simulations and in the performed test it achieved an average of 4.5%. Although the experimental results did not achieve the same result as the simulations, this is still deemed a very successful result as this was the first experimental test that succeeded in showing that it is possible to harvest energy from the wind field generated my a moving obstacle [28].

Another interesting takeaway from this paper is the relation between wind speed and throttle usage/. Here the relationship between the ground speed of the ship, the wind speed and the updraft velocities are tested well. It shows the importance of maintaining a good balance between obtaining enough potential energy to hover and enough kinetic energy to match the ground speed [28].

Next to ships, for the maritime case, it has been shown that the brown pelican has been able to soar by making use of the updraft created by waves [57]. This makes waves the only natural source available on the ocean for orographic soaring, which is all made possible thanks to the nature of the wave's own progression relate to the air, making it able to generate updrafts even if no wind is available [57]. The wave-slope soaring flight that is observed with Brown Pelicans. This behaviour is also seen in birds as albatrosses, fulmars, pelicas and gulls [40].

Herring gulls have been seen and known to follow big ships like ferry's and cruise ships to soar alongside [61].

What can be concluded from the research in static soaring is the difference of when, how and for what applications thermals and orographic lift is used. It should be clear now that orographic soaring is the better option for Micro Aerial Vehicles (MAV's) and UAV's, to get a clear overview, the main advantages and differences between orographic soaring and thermals have been listed in Table 2.3

Table 2.3: Comparison of the main features of Thermals and Orographic Lift and their differences

	Thermals	**Orographic Lift**
Availability	Low availability in Urban environments	High availibility
Predictablility	Low predictability	High, known to occur above certain objects
Strength	Weaker	Strong, at high windspeed
Size	Can span for km's	Can be big for Mountain's or ridges. Mostly "small".

2.3. Wind Fields

In the past section what the current knowledge and research is for soaring with UAV's. This section will work towards understanding the wind fields for orographic soaring and the theory behind it. Simulations and representations of the orographic soaring wind fields have been performed multiple times in past research [64][42]. This section will therefore give an overview of this research and how this will be of use for the thesis project.

Examples of the simulations and calculations of these different research cases will be shared, this will prove useful for generating realistic wind fields in simulation in the upcoming thesis project.

The results of [65][64] can also be used as validation as due to the similarity between normal shaped buildings, where it is won't really matter and the numerical results should resemble the ones for the simulated environment. The modelling of wind fields can be done in two different kind of ways, CFD and wind tunnel tests, both has been performed in the past.

2.3.1. Soaring conditions: Sink rate

The sink rate is one of the important factors for achieving soaring flights above buildings and in general [66]. The sink rate is the rate at which an unpowered UAV's,aircraft or bird descends [61]. This thus means to achieve soaring the updraft velocity should be the same or greater then the sink rate of the UAV's.

In [66], Watkins et al. concluded that the minimum sink rate in correspondence to the measured wind velocity is of much higher importance than the maximum lift-to-drag ratio for achieving soaring flight. Because of this, considering the glide polar of the airframe of the UAV's is important as it shows this relationship [58]. For the autonomous control of this, it is important to realize that at the maximum endurance speed, the sink rate will be the smallest/ [58].

During the wind-tunnel tests that were preformed, it was clearly seen that the sink rate varied between 0.5-2.6 m/s for wind velocities between 4.8 and 12.7m/s. This shows the exponential increase in sink rate in comparison to wind speed [66]. In the optimal soaring regions of the building the vertical wind component ranged between 15% & 50% of the measured airspeed while the stagnation point occurred at 80% of the building height. While other research into the wind fields around buildings confirms this [53].

2.3.2. Wind Fields around different objects

Although most urban environments will mostly have square shaped buildings for soaring, these shapes can vary. Therefore in this section different examples of wind field will be shown on all different kind of shapes. Here it will become clear how wind fields act differently for said shapes.

As discussed in subsection 2.2.2, extensive tests were performed on slope soaring for UAV's in [25], [58] and [28].

In [25], Hwang et al. show the wind field over a slope with a clear view of the viable soaring region. Because in this report the UAV's is autonomously looking for a hover position of 0% throttle, the soaring region is, as was made clear in subsection 2.3.1, the region where the sink rate is equal to the updraft velocity. A clear CFD analysis was performed and in Figure 2.2 the vertical upward velocity field can be seen. Where in Figure 2.3 the soaring region is shown as the region where there is no excess updraft.

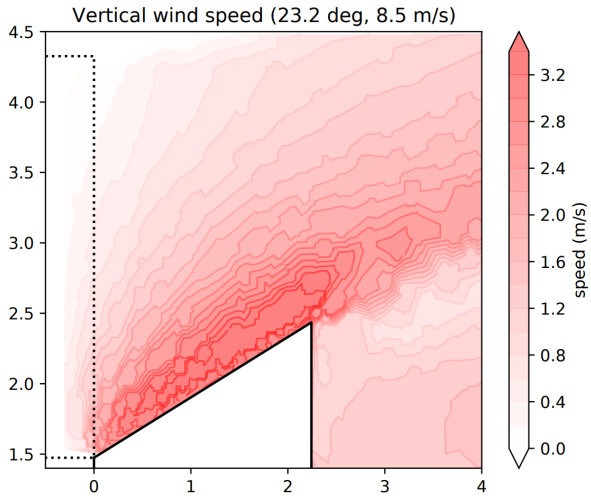


Figure 2.2: Slope soaring with the indicated vertical wind speed [25]

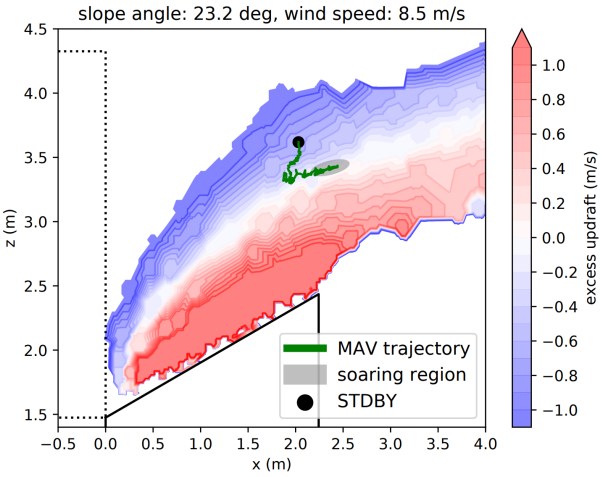


Figure 2.3: Slope soaring with the excess updraft windfield. Grey zone indicates the soaring region [25]

During the testing, the velocities as well as the slope angle were varied. Increasing the slope angle results in the soaring position moving to the front [25]. Both variations are best plotted on the glide polar as this shows the influence best. Both can be seen in respectively Figure 2.4 and Figure 2.5.

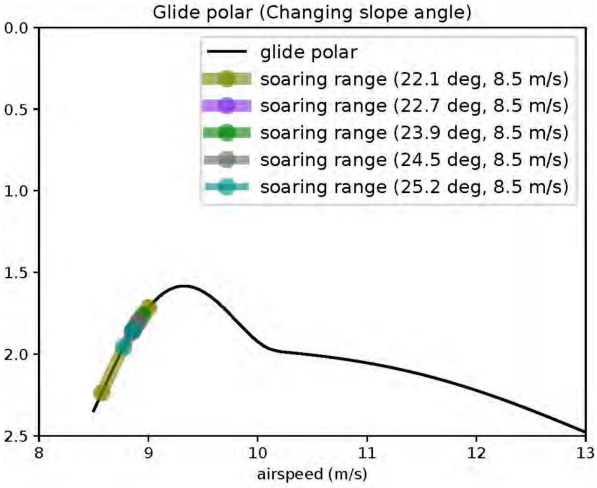


Figure 2.4: The change in soaring region when changing the slope angle [25]

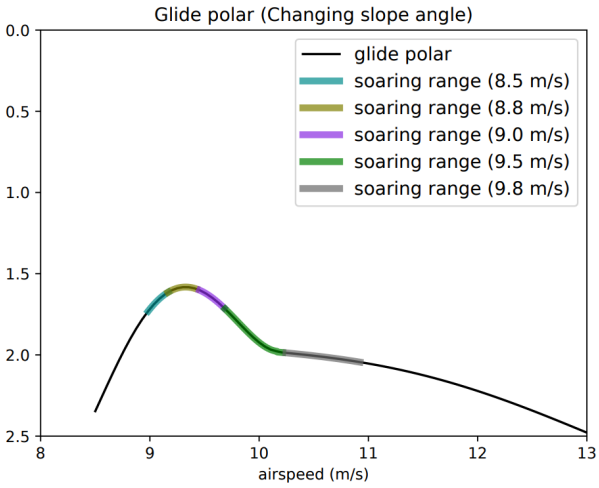


Figure 2.5: The change in soaring region while changing the windspeed [25]

Also from the MAVLab team at the TU Delft, a similar analysis was done for a cylinder shape by in [58]. In Figure 2.6 and Figure 2.7 3 different soaring positions are shown in the simulation and then on the glide polar. In Figure 2.7 the positions [a] and [c] have the same sinkrate while at position [b] the lowest sink rate is found.

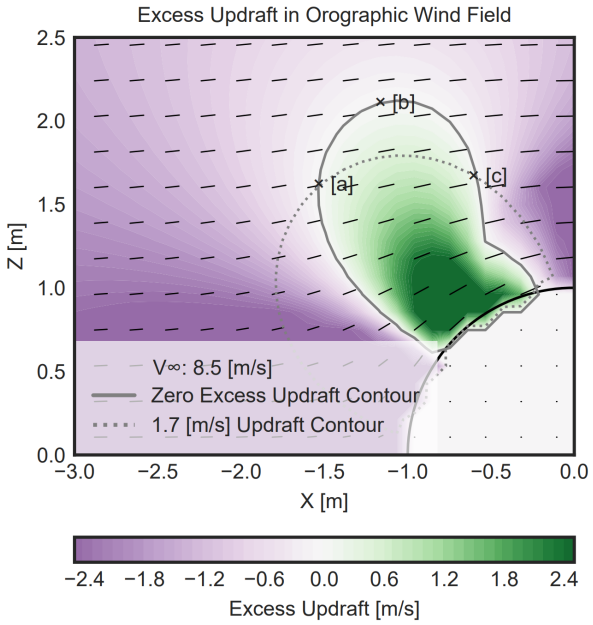


Figure 2.6: The excess updraft wind field on a cylinder shape plotted. [a], [b], [c] show different locations where soaring is possible on the zero excess updraft line [58].

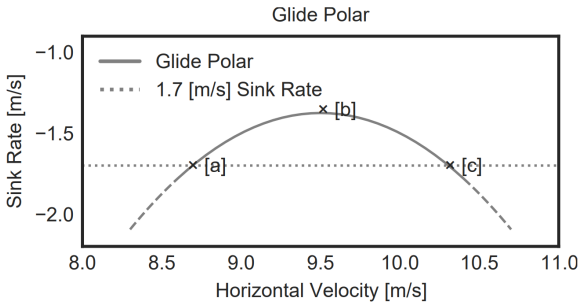


Figure 2.7: The glidepolar for the updraft windfield in Figure 2.6 [58]

The last non-rectangular shape that was studied by the MAVLab team at the TU Delft is the wind field from a moving object, in [28] this was a ship. As explained before in subsection 2.2.2 orographic soaring also occurs on moving objects like ships at sea. In Figure 2.8 a schematic overview of a performed CFD analysis shows how such a wind field looks. Here the wind field is dependent on 1 more variable than in the above discussed cases. Besides the wind speed and the shape of the object, the wind fields is now also dependent on the velocity of the ship, the moving object.

In [28] the sinkrate, which was explained in subsection 2.3.1, has not been used but here, but the effect of updrafts was modelled into differential equations. Because a moving object is being used, the relative motion of the UAV's has been put into equation form in Equation 2.2

$$\dot{z} = V_{TAS} \cdot sin(\gamma) + w_z \tag{2.2}$$

Now for the aircraft to use the updraft without using any throttle the following condition should be met [28]; The minimum rate of climb should be equal or higher than the minimum rate of climb, this is shown in Equation 2.3.

Equation 2.2 and Equation 2.3 show how all is dependent on the true airspeed, which is the relative velocity between the moving object and the wind velocity. Same can be said for the flight path angle, which is depended with the updraft which is again dependent on the relative velocity between wind and the moving object.

$$\dot{z} \geq 0 \Leftrightarrow V_{TAS} \cdot sin(\gamma) \geq -w_z \tag{2.3}$$

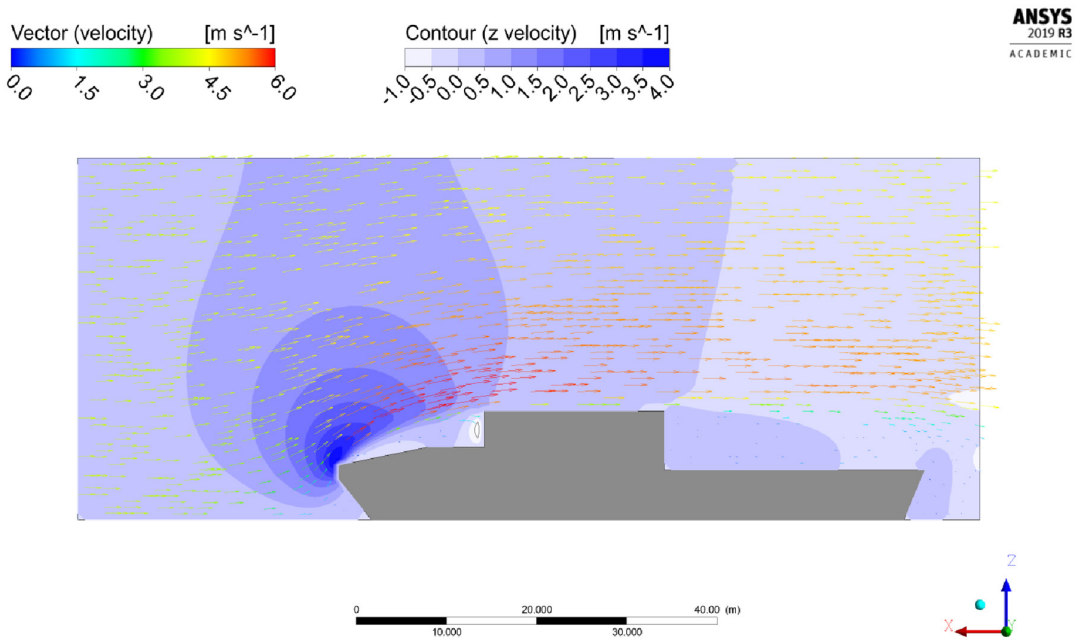


Figure 2.8: Schematic of CFD analysis and plotted wind field on a ship [28]

Another type of soaring structure that was discussed and also is of high importance for the research objective of the upcoming thesis project, is the wind field of buildings. As discussed earlier, [64], [65], [19] all performed CFD analysis on a selected building and pretty much came to the same conclusions considering the wind field.

In 2012 the research started of with analysing a scale model building inside the wind tunnel as shown in Figure 2.9, here the measurement grid was only in front of the building, which makes this analysis incomplete as the wind field also extends above the building.

In 2015 the research was continued and a full scale environment model was made of the campus. Making a full scale model accounts for the wind turbulence created by other buildings, this can be seen in Figure 2.10. In this figure the case of the updraft area above the building clearly shows the updraft region above the building, which could be important as in case of a crash, a crash on the roof would have less chance of hitting a human than crashing on the ground below.

A general note about urban orographic soaring is that the wind fields are always dependent on the wind direction and velocity more or less the same, which makes them quite predictable.

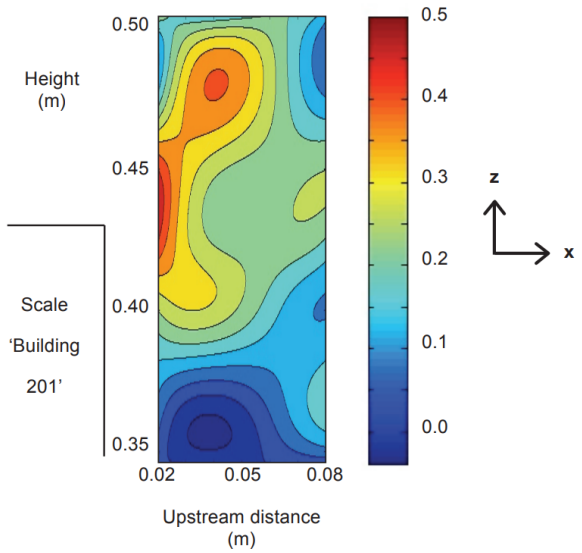


Figure 2.9: Vertical velocity as fraction of wind velocity on a building[66]

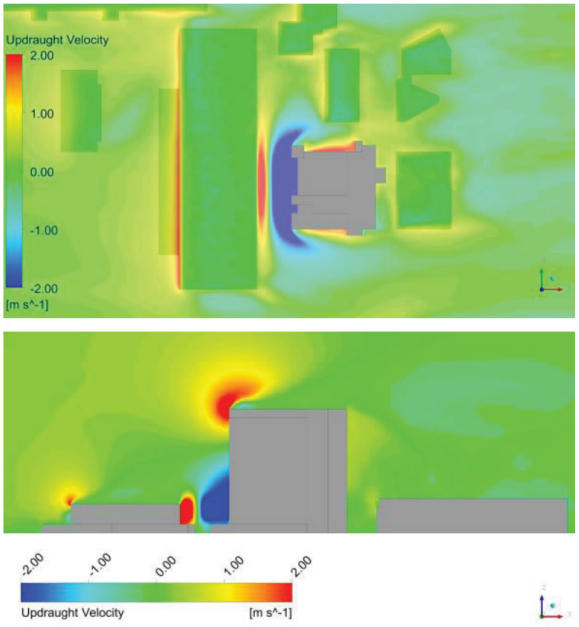


Figure 2.10: Updraft velocities in an enviromnet with multiple buildings [64]

2.4. Dynamic Soaring

As was just discussed in section 2.2, static soaring is bound to specific locations where soaring opportunities take place, whereas Dynamic soaring is dependent on the moving velocity at certain point in space or time [40]. Dynamic soaring also comes in two different techniques, as velocity difference in space is recognised to be gradient soaring while difference in time is called gust soaring. They have both in common that the wind field is not constant nor homogeneous and that they can be used for soaring [40].

From a biology perspective, albatrosses have been seen to use dynamic soaring the most. Here the albatrosses make use of the wind gradient, which is the difference in wind velocity at different heights. Lord Rayleigh mentioned this a first time already in 1883 in [47]. Gradient soaring, as said before, uses the difference in velocities at different heights. This difference in height is caused by frictional effects close to the ground which can also be other surfaces like waves and mountains. How higher from the surface the less friction thus this means that the wind speed will increase.

Dynamic soaring is a step by step process that is best described in the following way [49]:

Step 1 : The UAV's or bird positions itself within the slower moving wind.

Step 2 : The UAV's or bird climbs to a higher altitude, gaining potential energy, going back to Equation 2.1, and some airspeed due to the stronger wind.

Step 3: Now the UAV's should make a downward movement which converts the potential energy into kinetic energy.

Step 4: By now doing the last three steps on a loop, the vehicle will be able to use momentum to reach higher velocities and eventually higher gained altitude.

The schematic examples of this loop are given in Figure 2.11. Here the examples show dynamic soaring by birds using waves and hills.

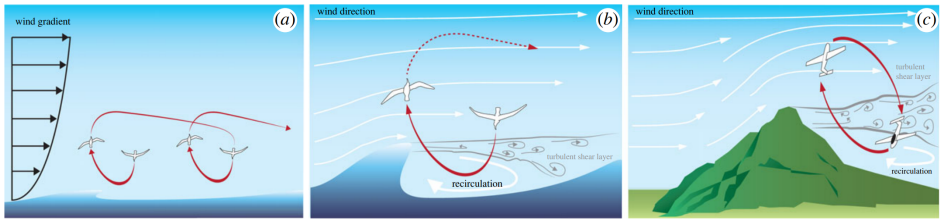


Figure 2.11: Schematic overview of gradient soaring at sea and using mountains [40].

Gust soaring, which is the other technique within dynamic soaring, uses sudden gust to gain atmospheric energy instead of the wind gradient. The mechanics of gust soaring is when a UAV's or bird uses the sudden wind bursts to have boosts of wind and therefore have increases in altitude. This is in a way similar to a surfer using the energy of waves to displace themselves [49]. Gust soaring is also much more stochastic then gradient soaring and therefore quite difficult to predict.

To conclude, gradient and gust soaring, thus dynamic soaring is an ingenious way for atmospheric energy harvesting. The main difference between the two techniques is that gusts and thus gust soaring is very stochastic while gradient soaring is more predictable and different techniques can be designed for this related to UAV's.

3

Machine Learning research

Machine Learning has now been common use for the application of autonomous UAV's. This chapter will first introduce how machine learning is applied to general cases of autonomous control. After this there will be given an overview of research that has been done on the autonomous detection of orographic soaring locations. Because as of this report, the research into this subject has been limited, research on different applications but with possible interesting techniques for the use case of this project will be given as well. It will be seen that there has been done a lot of research on autonomous UAV's that made use of the same inputs as for this problem. This chapter will be theoretical and should result in a selection of ML techniques that should be explored further in the thesis.

It should be remembered throughout the section that all data will be created by using simulator that is created for the thesis. Therefore all data will be self-created from the available environment, which is important to remember during the research into a suitable method.

The first ML technique that will be looked at is reinforcement learning

3.1. Reinforcement Learning

The general definition of Reinforcement Learning is that it is a branch of Machine Learning where an agent learns by interacting with a specific environment, this interaction is "graded" by a policy function, which should be maximized. One of the distinctions between RL and Supervised (Deep) Learning, which will be discussed in section 3.2 is that for the latter the model is explicitly taught what is correct and what's not whereas in RL it goes back to the concept of trial and error [1] [6].

3.1.1. The basics

To be clear on the terms that will be used in this section, the fundamental components of RL are listed below [1].

- **Agent:** This would be the drone, the learner
- **Environment:** All that the agent interacts with
- **State:** A complete description of the state of the environment, world.
- **Observation:** A partial description off the state
- **Action:** Action that is performed inside the environment by the agent. Every environment has its own set of actions, the set of valid actions within an environment is called the **action space**
- **Reward:** The reward, and therefore the reward function is of critical importance for the learning in RL. This function is dependent on the current state, action performed, and new state.
- **Policy:** The Policy is the equation that decides what action to take. It is said that the policy can be considered to be the agent's "brain". This policy is learned by mapping states to actions, which over time maximizes the expected cumulative reward. An important concept in RL is the challenge

of balancing exploration (trying new actions) with exploitation (sticking with known, rewarding actions).

3.1.2. Deep Reinforcement Learning Algorithms

As was stated before, the research into reinforcement learning has been extensive for years. Together with the advance of deep learning, neural networks have been integrated into reinforcement learning which has made way for Deep Reinforcement Learning (DRL) [6] [23].

Generally RL is categorized in two different types, model-free and model-based algorithms. The categorization is done based on the utilization of the environment models. Model-free methods do not have a model of the environment to help make decisions, these algorithms learn to make decisions by directly interacting with the environment [1].

Model-Based RL, as the name suggests, does use a model of the environment to make decisions. Here a model of the environment is learned first, this predicts the next state as well as the reward given a state and action. This model is then used to decide on what action to take. The main advantage to having a model-based algorithm is that the agent can plan by thinking ahead, this ensures a high sample efficiency [1].

In Figure 3.1 a taxonomy of RL algorithms is given. Here some of the used RL algorithms are given categorized under model-free & model-based.

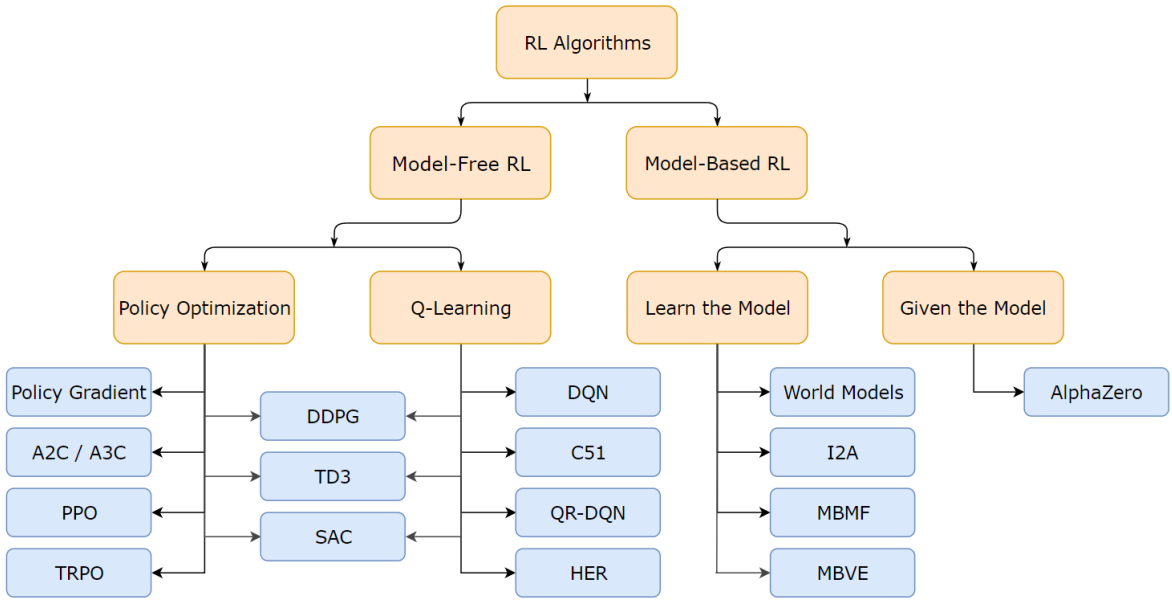


Figure 3.1: Taxonomy of Reinforcement Learning Algorithms [1]

3.1.3. Uses in UAV's

Although already mentioned in subsection 2.2.1, a deeper look is taken in the research on updraft recognition for UAV's. As mentioned, no research was found on orographic soaring recognition and the research was only focused on thermals and dynamic soaring.

In [43], the author takes on the challenging aspect of cross-country soaring. Here the challenge lies in the decision-making trade-offs between covering distance, exploiting updrafts, mapping the environment all while considering the unpredictable nature of thermal updrafts.

The example of the GPS triangle competition task is used with a model-free RL approach where the policy is only improved by observing the outcomes and the received rewards. As the agent knows all the locations of the thermals, this RL application does not cover any updraft localization but this is an optimal path planning application while making use of updrafts.

While other research also shows the tackling of the same optimal path planning problem while making use of thermals, none has shown more insight in detecting these thermals. While still recognizing this is something that should be research, the stochastic nature of thermals makes this difficult. For the detection of orographic updrafts, the same tactics as in [68], [43] and other RL techniques while tweaking the policy function for going to orographic soaring places in the urban environment. The fact that soaring locations will happen above the same places could make this a valid RL problem.

Other research that could be valuable when looking at their techniques were as followed. Researchers in the University of Zurich have looked into using DRL, here it was recognized that DRL has potential for solving time-optimal trajectory planning. This research was inspired from the work that was done by the Robotics and Perception Group at the University of Zurich for the AlphaPilot challenge. It should be noted that in the report, [20], no DRL techniques were used and their perception system was made using deep learning, something that will be delved further into in section 3.2.

With the end goal of this project in mind, the key takeaways from these two papers are that current research in DRL for autonomous drones is currently performed for trajectory-planning and perception is done with Deep Learning. This is not to say that once the updraft localization system is in place, DRL can be used for having route optimization by making use of updrafts, as what is suggested for further work in [43].

These key facts are further underlined by [59], here a target detection system for UAV's is being developed. Here it is recognized that for the object (target) detection Deep Learning is used. The other components like, path planning, navigation and control are being addressed with DRL [59].

The use of RL is an option for the upcoming thesis project, however in the following section a look is taken into using Deep Learning (DL).

3.2. Deep Learning in Autonomous UAV's

Deep Learning is currently one of the most researched subjects in the world, it has revolutionized computer vision and just recently a new revolution has started thanks to the advance in Natural Language Processing (NLP) and the creation of Large Language Models (LLM's).

In the beginning of the section the possibilities of DL are shown and the fact that DL has not been used as of yet for updraft recognition. Furthermore the section will delve into Supervised, semi-supervised and self-supervised learning, here a general overview of how it works as well as some research is shared. The research that is shared tries to find suitable tactics that could be used on the updraft recognition problem for the upcoming thesis project.

3.2.1. Updraft localization with Deep Learning

Deep Learning is a sub-field of Machine Learning that utilizes multi-layered artificial neural networks to extract and analyze data. Inspired by the human brain in both its functionality and structure [23], the term "deep" refers to the multiple layers of the network through which data is processed and transformed. The ability to process enormous amounts of data and continually refine the model by extracting features underpins the core principles of deep learning.

This literature study will give some insights on how different Deep Learning is currently used as well as find the most suitable method for the research objective. The section will furthermore provide the theory on the methods as well as reference some of the research that has been performed on it for the application of updraft localization. This could also be in the form of research on other applications that can be compared to updraft localization as, to the author's current best knowledge, there has been little to none research on updraft localization.

During the research into the different and right DL methods, it is important to look back at section 1.2 and set the constraints of the problem.

The subject DL goes synonymous with the concept of supervised learning. Supervised learning algorithms learn from a set of labeled training data [23]. The fact that it requires a large labeled dataset is for the current application not ideal. The potential training data will have to be created in the simulator, more about that in chapter 4. Although the artificially created data will try to be as accurate as possi-

ble, the fact that everything needs to be labeled is not ideal. Which moves the research into looking at algorithms that train on unlabeled data, this brings up two possibilities: Unsupervised Learning & Self-Supervised learning .

Unsupervised learning is a method that learns from an unlabeled dataset, with the aim to find structure in this data [23]. Some of the applications where it is used is for task like clustering and association [23].

3.2.2. Supervised Learning

The most popular way of deep learning is supervised learning, which uses algorithms to learn form a training dataset of labeled examples. This learning phase will provide the model with all kind of weights and the ability to generate an output from an input which is of the same type as the models training data [23]. This technique has been the most commonly used deep learning technique and is mostly used in image classification.

During training, the model of weights is asked to produce outputs, these outputs are compared to labeled data that was not in the training set. The predictions are compared with a loss function, where this loss function quantifies if the models predictions are good. Throughout the training the goal is to minimize the loss function, the most popular optimization technique for this is gradient descent [23].

3.2.3. Image classification with CNN

As discussed in subsection 2.2.2, the occurrence of updrafts in urban environments are quite predictable, as given a minimum wind velocity, certain objects like buildings, dunes, other human-made structure will have a wind field with vertical wind vectors. Due to this a simple classification neural network can be applied.

No specific works on just building classification has been performed due to the simplicity of the task. Some works on building features could be found however, this is already to much. A solution could be classifying building and using the area just above it with updraft regions, several building databases are available online for training. The classified images can also be aligned with the wind sensing sensor as the wind direction plays a role in the approach of the updraft wind fields.

An important part in supervised learning for computer vision has been the development of Convolutional Neural Networks (CNNs). The nature of CNNs lets them be controlled in depth and breadth thanks to which they are effective in tasks related to image perception [32]. The architecture of CNN's of which an example can be seen in Figure 3.2. CNNs have mainly three types of layers, convolutional layers, pooling layers and fully connected layers [63]. CNNs created the way for successful algorithms in computer vision applications like object detection, classification, robotics perception and self-driving cars [63].

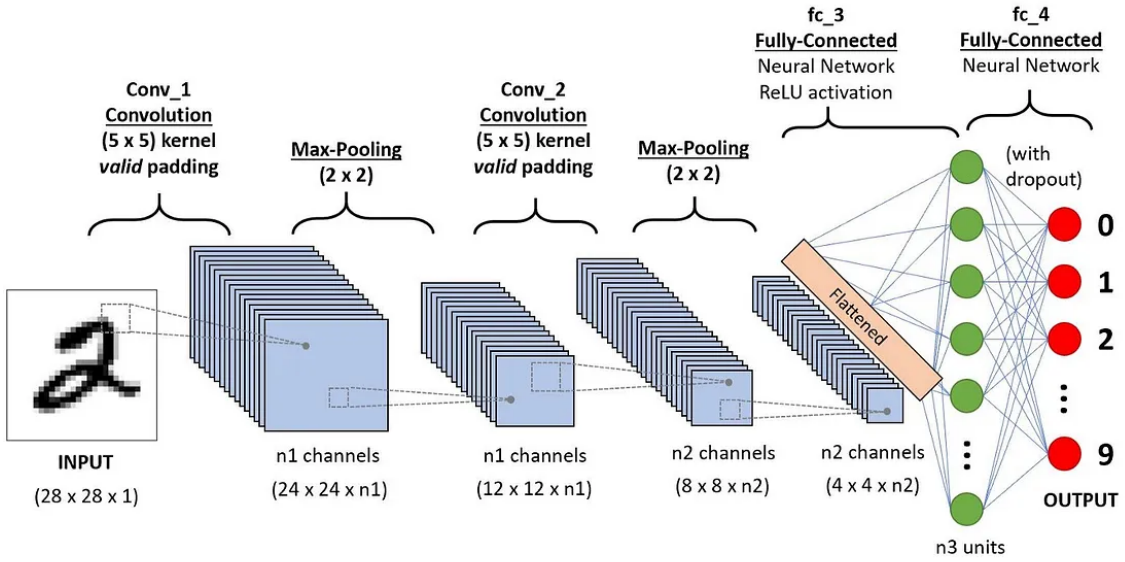


Figure 3.2: Schematic overview of CNN architecture for number classification from writing [16]

In relevance to the thesis project, the research on object detection has been extensive when considering object detection. Although there is no ”orographic updraft detection” network at the moment, the nature of orographic updrafts, as mentioned in chapter 2, makes it so that a network can be learned above which kind of structure these updrafts occur. Detectron2, is an object detection library created by Meta which houses all state of the art object detection networks which can be fine tuned [69].

One of the pre-trained models that could be used is one of the models from the ”R50”, where R50 refers to the ResNet-50 backbone architecture for this model [34]. The models in Table 3.1 are all state-of the art models available within Detectron2, all these models could be fine-tuned further for orographic soaring locations.

Taking R50-FPN, which is an R-CNN, would already give incredibly accurate masks while having the least computational power needed for training. R-CNN stands for region based convolutional network, which is an architecture specifically for object detection [22].

The ImageNet R50-FPN model has been compared between other has had state-of-the-art perforance while forming the back bone of other detection models like, Mask R-CNN, RetinaNet, Faster R-CNN and PointRend [29] [30].

In [29], Kirillov et al. shows the a version of the Mask R-CNN network with FPN (Feature Pyramid Network). Here the combination of the Mask R-CNN and the FPN network, which is a single-network baseline with the goal of having an accurate performance for instance and semantic segmentation. The model was trained and tested on the dataset Cityscapes, and had Average Precision (AP) values of 32 for the model.

The dataset Cityscapes, is a dataset that was made for the development of self-driving cars with thousands of street view images labeled [10]. This dataset could also be of use for the thesis project if it is decided to use supervised learning, as building recognition will then be of use. An example of how accurate the images in the dataset are labeled can be seen in **??**, where the building has a complex form but is still accurately labeled.

Kirillov et al. followed up their research in [30] where the design of PointRend takes place. PointRend is a neural network that has the ability to perform point based segmentation predictions. PointRend, just like Mask R-CNN FPN can be both used for instance and semantic segmentation. Due to the efficiency of the technique, the model is able to achieve an AP of 35.8 with R50-FPN as backbone on the dataset Cityscapes. This is better than PointRend while it should also be noted that due to the point based rendering of the model, the resolution is of less importance while this was for PointRend.

Figure 3.3: Labeled image from the cityscapes database [10]

Figure 3.4: Labeled image from the Cityscapes database[10]

While the previous papers trained networks to recognize buildings from the side or the front, for UAV's applications top view recognition would also be handy. In [70] a segmentation network based on the SegNet architecture was used to accurately recognize buildings from a top view perspective. The dataset that was used here, was their own however the authors also proposed the usage of dataset's like the Inria Aerial Image Dataset which contains many more labeled images [37]. Another option is to use available dataset's from the international society for photogrammetry and remote sensing [70].

While the previous research focused more on full masks and object detection, semantic edge detection presents a promising technique for building and thus updraft identification. Yu's work on this, in [73], demonstrates the potential it has when trained on a dataset like Cityscapes. The inherent nature of edge detection makes it appealing for integration with stereo or depth imagery. Doing this would possibly enhance the accuracy of the resulting spatial data, further helping in the process of updraft region detection.

However, a significant drawback of supervised learning is its requirement for enormous amounts of labeled data. In subsection 3.2.4 a deeper look will be taken into how the current research holds itself to combining labeled with unlabeled data in semi-supervised learning.

Table 3.1: COCO Instance Segmentation Baselines with Mask R-CNN [69]

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	mask AP	model id
R50-C4	1x	0.584	0.11	5.2	36.8	32.2	137259246
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525
R50-DC5	3x	0.47	0.076	6.5	40	35.9	137849551
R50-FPN	3x	0.261	0.043	3.4	41	37.2	137849600
X101-FPN	3x	0.69	0.103	7.2	44.3	39.5	139653917

3.2.4. Semi-Supervised Learning

Semi-Supervised learning sits between supervised, discussed in subsection 3.2.2, and self-supervised learning, discussed in subsection 3.2.5. Here the model is trained on both labeled data and unlabeled data, typically this means that only a small amount of labeled data is need while the biggest part of the training data is unlabeled [23]. Semi-Supervised learning therefore deals with the cost and time-consuming part of acquiring and labeling data. For the purposes of the thesis project this could be a viable option, as taking the idea of recognizing buildings, some images from a known labeled dataset could be used while it is completed by generated non-labeled data.

Important research was published Semi-Supervised object detection by Liu et al. in 2021. In [36], Liu et al. the make notice of the fact that most of the Semi-Supervised learning research has been focused on classification tasks neglecting object detection which has a higher labeling effort. Because of this they came up with the approach "unbiased teacher". Here the model trains two networks, a student and a teacher, where during training the teacher improves while guiding the student in the "learning" process". This model ensures that the pseudo-labels in semi-supervised learning are more accurate

and thus improving the overall technique. The unbiased teacher therefore showed an improvement of 6.8 AP more compared to the previous best method.

Semi-Supervised learning is a technique currently being researched for autonomous driving vechicles. Menke et al. developed a technique in [39] which leverages Semi-Supervised Learning to battle the cost of labeling data while achieving state of the art object detection performance. The technique that they've developed uses some labeled images and some "guessed" labels for training, they have combined this Semi-Supervised technique with "adversarial style transfer". Adversarial style transfer makes images from different domains look more like each other. The state-of-the-art network they've used for this is the transfer network AWADA. A transfer style network can best be looked as transferring the artistic style of a famous painting to a photograph. The reason why this technique is used is because the network is trained on synthetic data.

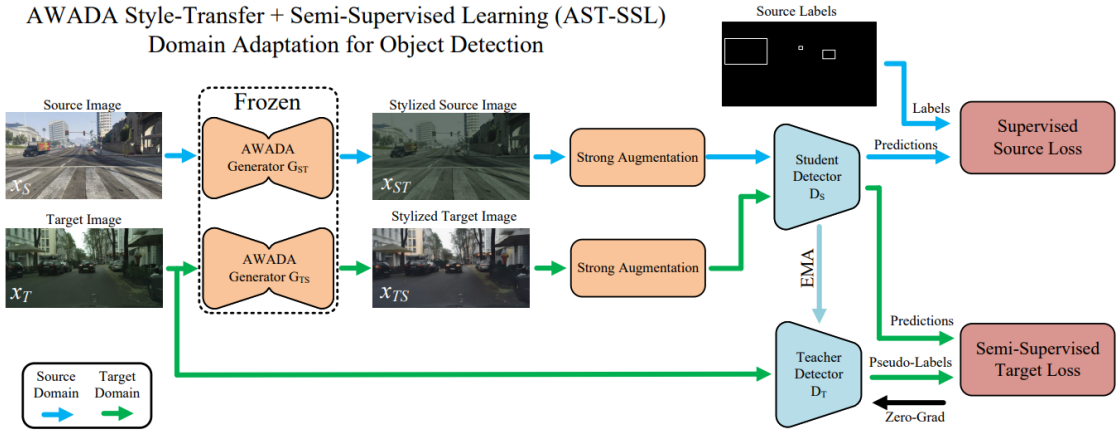


Figure 3.5: The AST-SSL network from [39]. The structure combines transfer learning with semi-supervised learning.

In this network, that is described in Figure 3.5, they combine style transfer technique with Semi-Supervised learning and create a more accurate network in the process. The network in this paper also uses the technique "Unbiased Teacher" developed by Liu et al. in [36] and was discussed earlier on. To try and understand the network the process goes as follows:

1. **Sampling of images:** The first step is sampling the source data and the target data. The target data does not have any labels as of yet.
2. **Style Transfer:** The style transfer technique, AWADA, will try to let both target and source images look alike.
3. **Student Network**: Images are processed by the student network. Which is the network that tries to detect the objects. The source images have ground truths, therefore the loss of the student network can be calculated.
4. **Pseudo-Labeling:** This is the Semi-Supervised part, here the target network does not have any labels so they let the "teacher network" of the model predict and assign these "pseudo" labels.
5. **Loss on pseudo labels:** The student network will now calculate the loss between the target images as well as the pseudo labels.
6. **Updating the models:** The student network is updated (trained). However, the teacher network is not directly updated. For this an exponential moving average is taken of the student network to update the teacher with that.

The paper concludes with having a higher AP then other state-of-the-art transfer style & semi-supervised learning networks when tested on the synthetic-to-real adaptation. This is also what makes this potentially interesting for the thesis project , as here a network trained on synthetic non-labeled data could prove useful.

[54] Provides another insight in how semi supervised learning can be used by delving into different data augmentation techniques with an emphasis on Generative Adversarial Networks (GANs) and Domain adaptation.

In the findings of the paper it is suggested that while traditional supervised approaches retain their superiority in data-rich environments, semi-supervised techniques with domain adaptation, emerge as alternatives when the annotating tasks are extensive. When compared to supervised learning methods, it showed that in situations with abundant high-quality ground truth data they still have an edge.

3.2.5. Self-Supervised learning

All solutions discussed in the previous sections have a need for some or complete data labeling. As the thesis project needs to set up a pipeline for others to simulate an environment where soaring as well as soaring detection is possible, a model that can be trained without annotating data would be a nice addition.

Here Self-Supervised Learning (SSL) can be of use, SSL is a Deep Learning method where a model learns to understand and extract meaningful representations, doing this with unlabeled data [71]. Having self-supervised data annotation is an important step as other methods require human annotated data which can be a limiting factor on the training process.

SSL could be a viable approach for updraft localization, as it has been proven to be very successful in dense localization tasks and object detection tasks [7]. The network should be set up by designing proxy tasks that leverage the available inputs such as vision, wind sensing, and depth sensing, of which training signals can be created that guide the drone to learn meaningful representations related to updrafts. SSL would allow the agent from unlabeled data, thus delete the time-consuming labeling process in the process.

Relevant work

Although a self-supervised approach has not been used for orographic updraft localization, this section will present relevant works which can be used to try and build out a strategy which can be used for the purposes of this project.

In [31] Kouris and Bouganis developed a SSL network which let the drone calculate the distance to collision with objects, which on its turn helped with the autonomous navigation of the drone to avoid obstacles. The Self-Supervised setup is interesting as here the depth sensors placed on the drone matches the measured distances to the visual data.

The input image was divided in three different regions on which every region was a distance was given for the closest distance to a certain object. This method could also be used for the thesis project as here the classification could just be binary in the sense that it is a soaring region or its not one. This shows the possibility of depth estimation with the usage of monocular vision, this way depth sensors/cameras are not needed to accurately calculate distances. The CNN network model was based on the AlexNet model, which is known to be a computationally low-demanding model.

Another way to approach this for the thesis project is that if the same network and strategy is used, the drone could be learned to estimate distances to all surrounding buildings, which together with the known wind direction and velocity could result in the drone localizing the closest viable soaring wind field.

Another self-supervised method with monocular vision was done by Li et al. in [35]. Here the approach is comparable to [31], with the biggest difference that the monocular vision approach matches other UAV's. An overview of the network output and the SSL approach can be seen in Figure 3.6. For the thesis project a similar approach can be taken where there is a image which is matched with the distance to the different objects inside the environment.

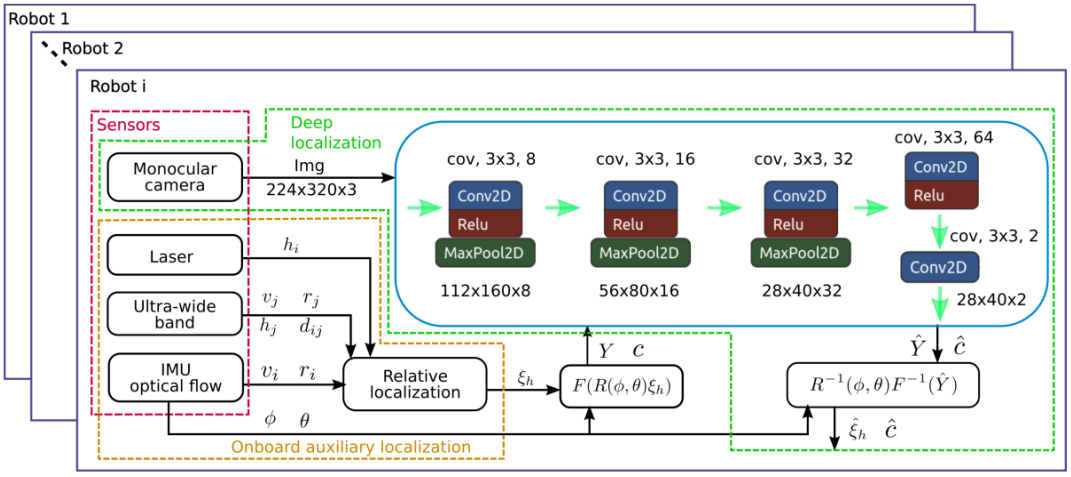


Figure 3.6: [35]

For this paper an simulation pipeline was setup. This was completely done using Blender. Using Blender it became possible to render multiple realistic 3D robot models. Important here is that the positions and depth of the robots are known in the rendering environment. Because of this the data from Blender can be used as ground truth data to create the labels for the synthetic dataset. More about the use of Blender in chapter 4.

Already in subsection 3.2.2 it was shown how depth images could work as a good solution for detecting updrafts. For Self-Supervised learning some research has been performed in monocular depth estimation. While [24] and [46] focus on depth estimation for autonomous driving, [18] looks into depth estimation for UAV's. Although very interesting and a good showcase of the power of self-supervised learning, it has not been clear how depth estimation would help in the localization of updraft regions.

3.2.6. SSL for the thesis project

Utilizing a simulation environment makes the application of SSL an attractive avenue for the thesis project. Though the specifics of its implementation remain to be fleshed out, the preliminary research provides a compelling basis for the author to delve deeper into this area and explore the potential benefits.

Should SSL prove to be overly complex, transitioning to semi-supervised or even supervised learning remains on the table. Nevertheless, bearing in mind the ultimate objective of devising a simulation pipeline for drone training across varied environments, the prospect of labeling every new dataset becomes a significant drawback.

4

Simulator Research

To achieve the research objective a simulator already available will have to be chosen to create the environment that will be needed. This chapter will explore the trade-off between current off-the-shelf simulators for UAV's. Furthermore it will also clearly set the requirements for this simulator as well as look into the urban environment creation within the simulator.

The chapter will give an overview of the research that has been performed in the field as well as current ideas that were realized during other research.

4.1. Pre-Requisites

The search for the right simulator for the application described in this paper as well as for the further thesis research, is done with setting a list of "needs". Ideally the simulator would be able to comply with the needs of the thesis goals set in section 1.2.

Because the localization will be done by training an agent, this AI will need inputs. These inputs therefore also need to be available in the simulation environment. Some inputs that need to be kept in mind when looking at software are:

- Depth Sensing
- Wind direction sensing
- Visual cues

Next to this, if visual cues are to be used, the realism gap between the simulator and real-life should be minimal. This is only when the agent relies on visual cues, it could be that depth sensing alone is enough. Therefore some extra requirements are set when looking at a simulator:

- The simulation rendering should be realistic, minimal simulation-to-reality gap
- Availability of API for ML
- Minimal computational effort
- Availability of real life sensors: Wind sensors, depth sensors, cameras, ...
- Ability to create and store synthetic data

4.2. Environments and CFD

1. **Blender**

   For the environment creation one of the most used 3D computer graphics software that is being used. Blender is a great rendering tool where its use in simulations is perfect for the current application [8]. Another important functionality from Blender is its python API where its functionality can be expanded using Python scripts, making it possible to write custom tools if necessary [8].

The last important quality of Blender is its easiness to import and export to multiple data files [8][13].

In [13] an example is done of how to create a real life environment in Blender, for this example the city of Brussels was used. This example can be seen in Figure 4.1.

Blender makes it possible to import terrain models from the ”Urbis dataplatform”, which opens up the possibility for doing the same for example the TU Delft campus, hovering above the Aerospace faculty for example [13].

In [14] the author takes the environment creation a step further. Here, using the Blender add-on ”SketchUp” is used to import the textures of the buildings for the 3D model. Having these textures is important for the realism of the agent, when computer vision is used. The example that was shown in the article can be seen in Figure 4.2

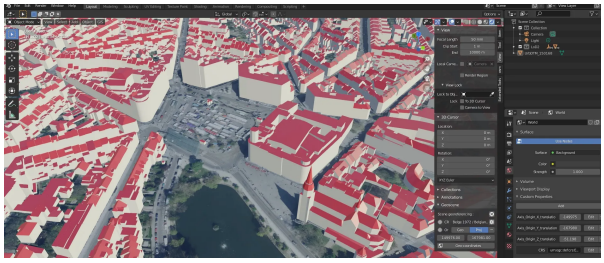


Figure 4.1: Real life buildings created in blender

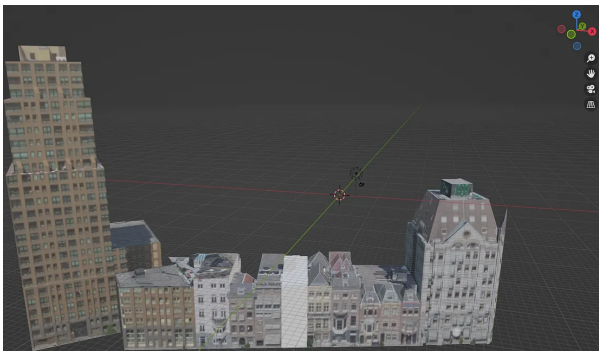


Figure 4.2: Rotterdam buildings in Blender

The following step should be performing a CFD analysis on the created environment, something which was done by Nathanael et al. in [42]. In this report a realistically rendered environment of a neighborhood in Singapore was created in the same way as the example described earlier. After this environment creation, a CFD analysis using OpenFOAM was performed with the focus on the wind turbulence. For the case of the upcoming thesis project, the same process should be repeated but with focus on the vertical wind field.

Another interesting framework that has been released recently is the open source framework ”WorldGen” [55]. WorldGen is a generative simulator which means it can generate countless different environments with very realistic textures [55]. This creation of synthetic data would help a lot in the training of an agent and would be a nice addition in comparison to making certain environments yourself. The code repository however is as of this date still not made public so it can not be used, maybe in the future.

2. **OpenFOAM**

   Multiple reports suggest that for the wind flow simulation of urban/city environments as well as single building CFD simulations OpenFOAM is more than suitable [17], [42]. The fact that it is open source and easy use-able for Blender created environments is another advantage [21]. In short the process using OpenFOAM would look like the following:

   (a) Import from Blender

   (b) Mesh generation

   - blockMesh
   - snappyHexMesh –> More appropriate for complex urban geometries.

   (c) Setting up the simulation

   - Defining boundary conditions
   - Specifying the turbulence model: Choice between, Reynolds Averaged Navier Stokes (RANS), Large Eddy Simulation (LES) & Detached Eddy Simulation (DES). RANS can

be used as it is less computationally expensive than the others and this was also used in [42]. When considering a single building, LES could be used as it is more accurate, however for big urban environments it gets computationally too demanding [17].

(d) Running the simulation

(e) Post-Processing

- paraView: Package for visualizing the flow patterns, wind fields and more

Lastly, this CFD will have to be integrated with the simulator, which is chosen in section 4.3. Depending on the simulator there will be the possibility for offline analysis or online coupling. However, it is likely that the CFD will be done at first and than the simulations will be encoded in the simulator as having the simulator coupled to the CFD solver for real-time interactions will be too computationally demanding.

4.3. Simulator: Pegasus Simulator

Throughout the years multiple (aerial) simulators have been developed, in Table 4.1 the most relevant simulators are given. The table also gives an overview of the simulator requirements that were set in section 4.1. The simulator that is selected for the thesis project is the Isaac Gym, this section will further explain how this choice was made.

At first Flightmare seemed the best option together with AvoidBench. AvoidBench is an extension made upon Flightmare. RotorS and Hector do not compare to flightmare in Sensors as well as in ML API. AirSim was seriously considered as well, however here the ML API came in the form of a wrapper and for Flightmare this was build in.

Table 4.1: Simulator comparison

Simulator	Rendering Engine	Sensors	ML API	Physics Simulations
RotorS	OpenGL	IMU, RGB	No	CPU
Hector	OpenGL	IMU,RGB, Depth	No	CPU
AirSim [52]	Unreal	IMU,RGB,Depth, Segmentation	No/Yes(wrapper)	CPU
Flightmare [56]	Unity	IMU, RGB, Depth, Segmentation	Yes	CPU
AvoidBench [72]	Unity	IMU, RGB, Depth, Segmentation	Yes	CPU
Isaac Sim [38]	Nvidia Omniverse Platform. RTX Renderer	IMU, RGB, Depth, Segmentation	Yes	GPU

Now ISAAC Sim is created by a team at NVIDIA, where they specifically created this environment specifically for reinforcement learning research [33][15]. The biggest difference with the other simulators is the fact that the physics simulations are done by use of GPU. This is an important difference as the researchers claim that using GPU gave around 3 orders of magnitude improvement compared to simulators using CPU [38]. ISAAC Sim has some built in environments and agent dynamics, however none of use for the upcoming thesis project. ISAAC Sim is also created with the process of SSL in mind. The simulator is designed so that it is very easy to generate Synthetic Data, this feature would be a big advantage for the upcoming project.

Now in May 2023 Kulkarni et al. came up with a quadrotor ISAAC Sim build called Aerial Gym [33]. This version only has quadrotor dynamics, which means that for the purpose of the thesis project, an Aerial ”fixed-wing” Gym will have to be created upon ISAAC Sim. Hoping that the work in [33] can help with this.

Now another framework that has been build upon ISAAC Sim is the Pegasus Simulator [26]. Jacito only recently published their work in July 2023, making this simulator the newest of all the related work. The simulators discussed and listed in Table 4.1 have all one thing in common, which is they are built with a game engine that is not specifically focused on robotics applications, however Pegasus Simulator is built this way as it is built on top of the NVIDIA Omniverse suite.

The design of the Pegasus simulator was done with keeping in mind the best features of the most used UAV's simulator frameworks. Designed as an Python extension on the NVIDIA ISAAC simulator, a gazebo like experience was created but with much more detailed rendering [26]. In Figure 4.3 a schematic of the Pegasus framework can be seen. Within this framework the author states that it is possible to create and use other vehicles than a quadrotor, which also can have its custom sensors [26]. As this is really the first time that the agent configurability is so clearly stated, the decision of using the Pegasus Simulator framework for the thesis project is even more substantiated.

The hope is that a fixed-wing simulator framework can be built upon the Pegasus Simulator. Combining this with a pipeline to easily merge the environment and the CFD generated wind fields, the ISAAC Sim and the Pegasus Simulator extension seem far-out the best possible option to use as simulator framework.

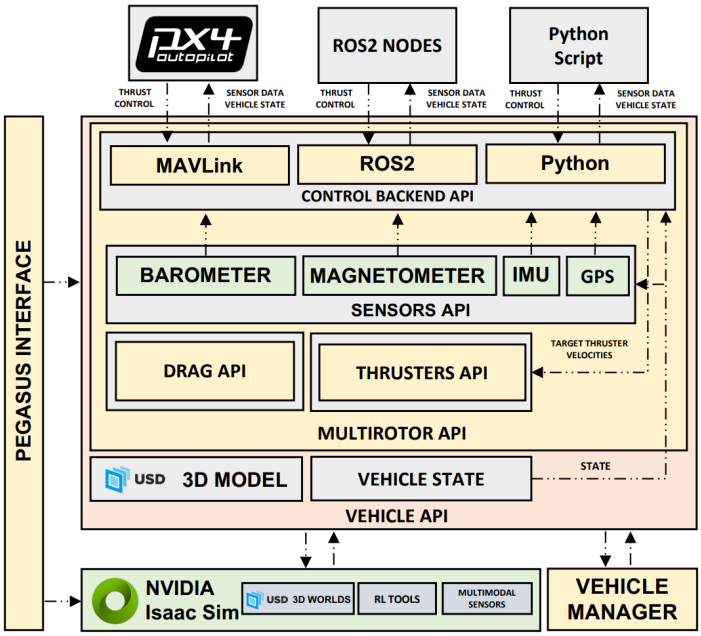


Figure 4.3: Schematic overview of the Pegasus Simulator framework built upon ISAAC Sim

5

Thoughts on thesis

5.1. The research gap

In this section the gaps in the current state-of-the-art research are explained. The gap in the research is what the upcoming thesis project will try to fill with answering the research question established in chapter 1.

From chapter 2 the current research in all the soaring techniques are discussed, these soaring techniques are well researched. It has been generally recognised that use of soaring techniques for atmospheric energy harvesting is a viable option for energy/endurance limited UAV's.

chapter 3 delves deeper into the research in ML for the thesis , a clear gap in the research occurs. While there's research on detecting objects or gauging depth using various ML techniques, the specific domain of orographic updraft recognition remains largely untouched. The research is therefore focused on analog style papers of which the discussed ML techniques could potentially be translated in to orographic updraft recognition.

Consequently a review has been done of the current state of the art techniques, that could be used. Historically ML models have been anchored heavily in supervised learning, requiring significant amounts of labeled data. In contrast, self-supervised learning, which is newer, offers a compelling alternative. What should become clear during the thesis project is how the advantage of losing the need for a time consuming annotation process weighs up against the challenge of it being a newer technique as well as being compared to supervised accuracy. Nestled between these two methodologies is Semi-Supervised learning, which might bridge the best elements of both.

As all different techniques are explored, a decision will have to be made between the tried-and-tested annotating intensive supervised learning or the newer, more intricate self-supervised learning which shows a lot of potential. Deciding on the right technique to used will play a vital role in filling the research gap.

It's worth noting that from the research into the behavior and occurrence of orographic soaring wind fields, showed that they are very predictable under known wind conditions. For instance, given a certain wind speed, a building will consistently produce a similar orographic wind field. The predictability of these wind fields could play a valuable role to designing a strategy and tactic to fill the main research gap.

The last thing needed to be researched was the current state of UAV's simulators. Here chapter 4 showed how different state-of-the-art simulators are available, however a gap exists in merging detailed CFD wind fields with the simulator. Another gap that has been identified is the lack of fixed-wing UAV's simulators that currently exist. All the viable simulators found did not allow for fixed-wing configurations and were only designed for quad rotor dynamics. This might be attributed to the fact that a lot of commercial UAV's simulators are tailored for quadrotor dynamics. Additionally, a seamless configuration between the chosen ML technique and the simulator has yet to be addressed and currently

poses as another gap.

This report shows the research that has been done in updraft detection/sensing, as well as autonomous control inside or close to updraft regions. It also showed theory, tactics and algorithms for optimal path planning by making use of updrafts. All these cases rely on previous knowledge of updraft regions, or for the case of thermals after first training a prediction model and then using the predictions.

5.2. Planning

This section will go over the planning for the thesis project. As discussed in section 1.2 the primary and subsidiary research goals for the thesis project have been identified. These goals are now scheduled in for the coming months.

The three core goals mentioned in chapter 1 and section 5.1 simplify the planning process. The general planning is first setting up the environment and the simulator, then implement the Machine Learning algorithm. This is finished by a month of reporting the findings, although it is advised and planned to report some of the techniques found and used, during the implementation of said techniques.

For the first main block, which is setting up the simulator and the simulator environment. This block has been assigned around two months and should be done around the middle of October. Although this could take quite some work and setting up environments could take up months, most of the environment set up and simulator setup has been done during the literature study, so there is a strong possibility that this is done around the end of September.

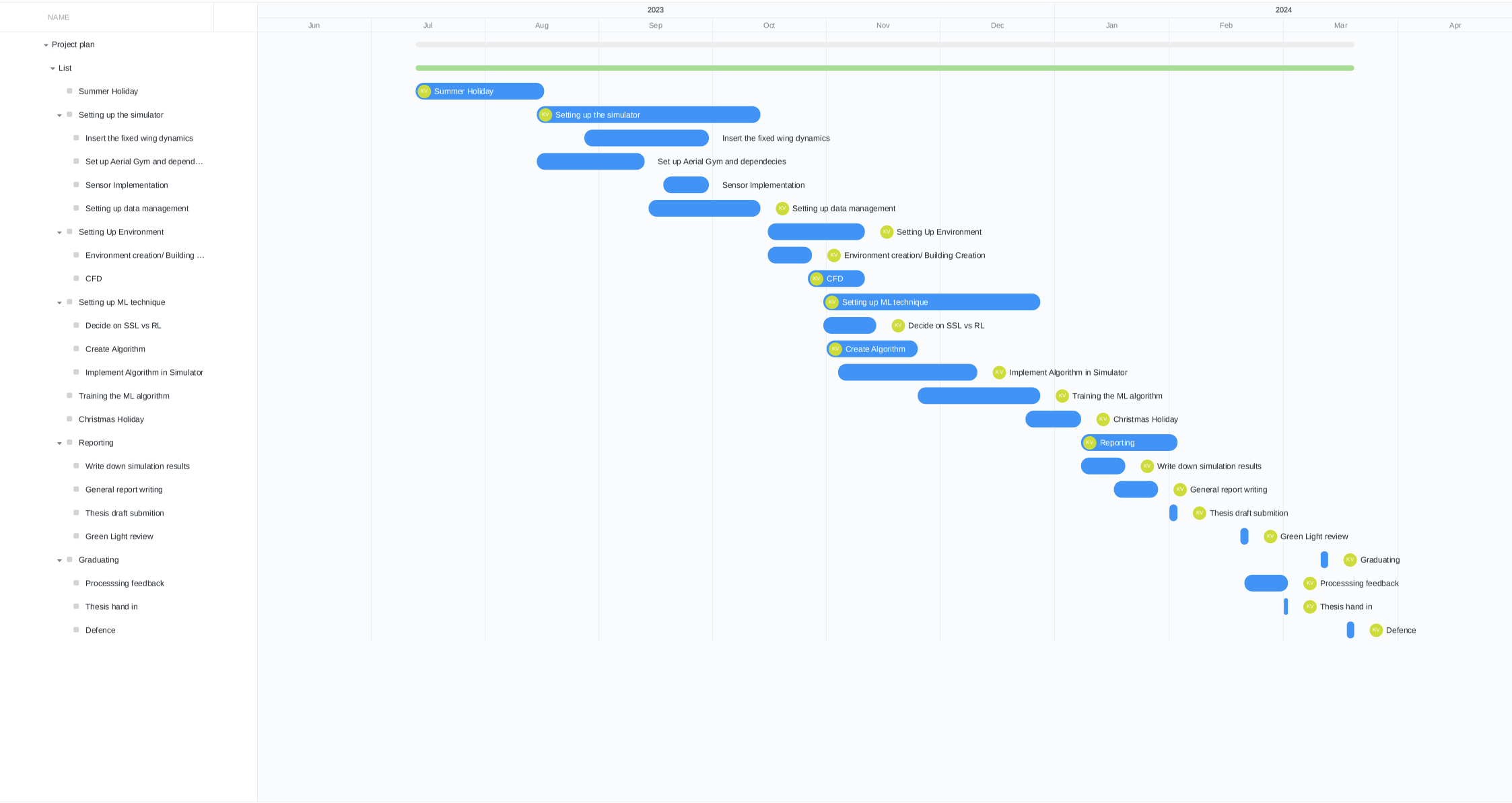
Furthermore a month has been assigned to setting up the environment. Setting up the environment consists of the following two main tasks, the urban environment creation and the CFD analysis on this environment. How the CFD will be performed as well as the needed theory for this was performed during the literature study and is described in section 2.3 & chapter 4.

For the Machine Learning part of the thesis, first a decision will have to be made on what ML technique to use, some possible techniques have been described in chapter 3. However as was mentioned, the most suitable one will have to be discovered while reading and trying more. This is also the most difficult part of the thesis as finding and creating a suitable algorithm is the main part of the research gap, as described in section 5.1.

Lastly, the agent will have to be trained inside the environment. For this a month has been scheduled, which means that if everything goes according to the project plan, the product should be done by the end of December. After the Christmas break a month is foreseen for reporting.

The graduation process, will start off with an official draft submission by the end of January and, following the dates shared by the TU Delft [2]. This would mean the green light review would happen 2 weeks after the draft submission around the 20th of February. After the feedback is processed, the thesis hand in is planned around the first of March, which given the minimum of 20 working days would result in the thesis defence on the 20th of March.

It should be accounted for that, as this project is mainly a software project, the unforeseeable nature of bugs and library issues can cause a planning to differ. Hence the use of sprint sessions in most software companies. But nevertheless this planning seems reasonable and will increase efficiency and give structure by having set different deadlines for the different parts of the thesis.


NAME
2023
2024
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Project plan
List
Summer Holiday
Setting up the simulator
Insert the fixed wing dynamics
Set up Aerial Gym and depend…
Sensor Implementation
Setting up data management
Setting Up Environment
Environment creation/ Building …
CFD
Setting up ML technique
Decide on SSL vs RL
Create Algorithm
Implement Algorithm in Simulator
Training the ML algorithm
Christmas Holiday
Reporting
Write down simulation results
General report writing
Thesis draft submition
Green Light review
Graduating
Processsing feedback
Thesis hand in
Defence
KV
Summer Holiday
Setting up the simulator
Insert the fixed wing dynamics
Set up Aerial Gym and dependecies
Sensor Implementation
Setting up data management
Setting Up Environment
Environment creation/ Building Creation
CFD
Setting up ML technique
Decide on SSL vs RL
Create Algorithm
Implement Algorithm in Simulator
Training the ML algorithm
Christmas Holiday
Reporting
Write down simulation results
General report writing
Thesis draft submition
Green Light review
Graduating
Processsing feedback
Thesis hand in
Defence

6

Conclusion

Building on the insights that were gained during this Literature Study it has become clear that building further on soaring techniques is one of the ways that will help in the innovation race to solve the endurance problem for UAV. The study shows clearly the importance as well as the different advanced soaring techniques that have been studied as well as the ones that need some more research. The predictable patterns of orographic wind fields form a good base for the research in the upcoming thesis project.

The exploration into ML techniques shows the role AI has played in the field of UAV's as well as the role it could play in the advancement of soaring techniques, and in the case of the upcoming literature project, orographic soaring detection. The comparison between supervised, semi-supervised and self-supervised learning models offers valuable insights into their strengths and limitations. The potential of SSL in particular emerged as the most promising as this way the problem of the labor intensive data labeling process can be avoided. The shift to more efficient learning models has also been coming up the recent years in other fields.

The study also spends a chapter on the possible current state-of-the-art simulators that can be used to integrate said ML techniques. Here NVIDIA's Isaac-Sim and its Pegasus-Simulator extension show the most promise. They present the unique opportunity to develop a pipeline which can simulate orographic wind fields, a fixed-wing UAV's as well as create synthetic data for Self-Supervised learning. The creation of complex urban environments is also possible with the use of Blender, these environments can then be imported to the the simulator.

Finally, the key research gaps are identified. The study clearly shows the gap in machine learning applications on the detection of orographic soaring locations. Furthermore it clearly shows the lack of integration between realistic wind fields and simulators as well as a lack in fixed-wing UAV simulators. Addressing these gaps will be essential for answering the research question in the upcoming thesis project.

Part III

Software Structure & Results

7

Software Functionality

This chapter will give a more extensive explanation of the software architecture of the Osprey Simulator, and more specifically the engine Isaac Sim, than was given in the scientific paper in I. It will focus on the use of USD files, as this was a steep learning curve during the thesis. This chapter is therefore written to further easen reproducibility.

7.1. Overview of Isaac Sim

As discussed previously in the literature study and scientific paper, Part I & chapter 4, Isaac Sim is an advanced robotics simulation platform developed by NVIDIA and built on the Omniverse framework [38]. It offers the opportunity to simulate high-fidelity, real-time environments mainly for the use and testing of robotic systems, sensor integration, and autonomous algorithms. Isaac Sim is widely recognized for its seamless integration with machine learning workflows, physics-based simulation, and a robust application programming interface (application programming interface (API)) that supports extensive customization. These capabilities make it an ideal tool for both research and industrial applications in robotics [38]. Isaac Sim has been used as the engine for this thesis project, where fixed-wing dynamics and an atmosphere have been built upon it.

Although Isaac Sim has great functionality, some more explanation is needed, especially for the use and management of USD files, which is done throughout the simulation for every object.

7.1.1. The Role of USD Files in Isaac Sim

Universal Scene Description (USD) files form the backbone of Isaac Sim's scene management. USD is an open, extensible framework developed by Pixar for 3D scene description, and it is utilized in Isaac Sim for the following reasons [38, 12]:

- Scene Definition and Composition: USD files allow for complex environments, including geometry, materials, lighting, and hierarchical relationships among objects.
- Hierarchical Structuring: With USD's inherent support for scene graphs, complex assemblies (such as multi-jointed robots) can be organized into logically related components. This hierarchy simplifies the management of articulation points and sensor placements.
- Live Updates and Collaboration: USD supports non-destructive editing, allowing real-time changes to objects and environments that facilitate iterative design and collaborative development. For example, during testing the mass of the drone can be easily changed.

  The use of USD files not only streamlines scene creation but also enhances interoperability with other digital content creation tools.

7.1.2. Understanding Articulation Points in Isaac Sim

One of the more challenging yet critical concepts in the Isaac Sim simulations was that of articulation points. These are essentially the joints or connectors that dictate how various parts of a robotic assem-

bly move relative to one another. For the purposes of the thesis project, there was only one articulation point around the center of pressure of the wing. However, in more extensive simulations, the possibility of creating control surfaces and connecting these through articulation points offers a lot of opportunity for further research with Osprey Simulator.

The Articulation Points in Isaac Sim, which represent the joints (revolute, prismatic, spherical, etc.) connect the different components of a robot [11]. They are fundamental in determining the degrees of freedom and movement constraints within the simulation. The correct modeling of articulation points is essential for accurately computing the kinematics and dynamics of connected bodies. This involves complex mathematical formulations and careful tuning to maintain simulation stability and realism.

Each additional joint introduces further variables that affect system behavior. Misconfigured articulation points can lead to numerical instability or unrealistic behavior in simulations. It is therefore advisable to test individual joints and constraints in isolation before integrating them into the full robotic system. Although Isaac Sim offers tools to visualize and debug joint behavior, which can ease the process, it stays an error-prone but important process.

7.1.3. The Built-in Sensor Suite and Its Impact on Osprey Simulator

Isaac Sim's comprehensive, built-in sensor suite has played a pivotal role in the development of the Osprey Simulator. The sensor suite includes detailed and configurable models for cameras, LiDAR, inertial measurement units (IMUs), and other sensors commonly used in robotics and autonomous systems.

By leveraging the realistic data output from Isaac Sim's sensor models, the Osprey Simulator could integrate and validate sensor fusion algorithms more effectively. This will also at some point help with the transition to real-world sensor data and usage.

7.1.4. The Synthetic Data Generation Tools

Another significant feature of Isaac Sim is its built-in synthetic data generation capabilities. These tools enable the capture of diverse data types such as RGB images, depth maps, and point clouds, which are all crucial for training and validating autonomous systems.

Isaac Sim's synthetic data generation tools provided the foundational methods for capturing detailed depth and point cloud data. Although extensive work was needed to refine these outputs for the specific localization of updraft application, the built-in methods served as the initial basis for data collection. The ability to automatically generate large datasets under controlled conditions allowed for building extensive training datasets, and leaves open the possibility to scale this up if the available computational power allows it. By integrating synthetic data generation directly within the simulation environment, it becomes easier and also helps to close the gap to real-life tests.

7.2. Closing remarks on Isaac Sim

Isaac Sim stands out as a powerful simulation platform that not only provides a realistic environment for robotics testing but also offers advanced tools that have been instrumental in projects such as the Osprey Simulator. The integration of USD files for scene management, the careful modeling of articulation points, and the extensive built-in sensor and synthetic data generation capabilities have all contributed to its success in supporting complex simulation tasks. These features have now set up the Osprey Simulator to eventually move from simulation to real-world applications.

For more information about the installation of Isaac Sim and Osprey Simulator, the Readme of the source code project can give more information. Here the other requirements needed to run as well as some installation tips and USD environment tips are mentioned. The code can be found in Appendix A

8

Soaring Spot Detection Network Results

8.1. Testing and Results

After the training, described in the scientific paper in Part I, the model's performance was evaluated on a separate test dataset comprising depth images and wind vectors that were not used during training. The inference dataset consisted of the TU Delft environment as well as the Random Environment. Overall, the model demonstrates strong generalization capabilities and accurately predicts updraft regions across varied environmental scenarios. This section presents several hand-picked results that show the behavior and illustrate both the strengths and limitations of the model.

Notably, in the randomly generated environment, characterized by simpler geometries, the model produces precise predictions, whereas in the more complex TU Delft campus environment, certain challenges arise due to intricate architectural details, which was to be expected. This difference can be seen clearly in Figures 8.1, 8.2, 8.3, 8.5 and 8.7. This is not unexpected behavior as, he updraft and wind flow in simpler environments tend to be less turbulent and more coherent, disrupted by orographic structures with general shapes like cylinders, rectangles, triangles etc. While for the real-life environments, due to all the complex shapes in there, the wind field exhibits more unexpected behavior.

The model, also shows great context awareness, as can be seen in Figures 8.1, 8.2, 8.3, 8.4. It accurately predicts when there are downdraft as well as when there are updrafts. This is an important capability, since updrafts typically occur only on the windward side of structures, where the wind impacts the building. Recognizing this pattern is vital for future navigation towards soaring hotspots, ensuring that the model does not erroneously predict updrafts on every rooftop. These results indicate that the model effectively interprets wind direction and its relation to updraft localization from the wind vector input.

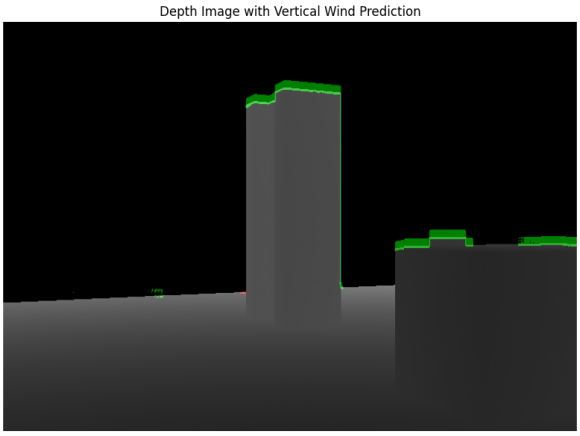


Figure 8.1: Updraft Prediction, Random Environment

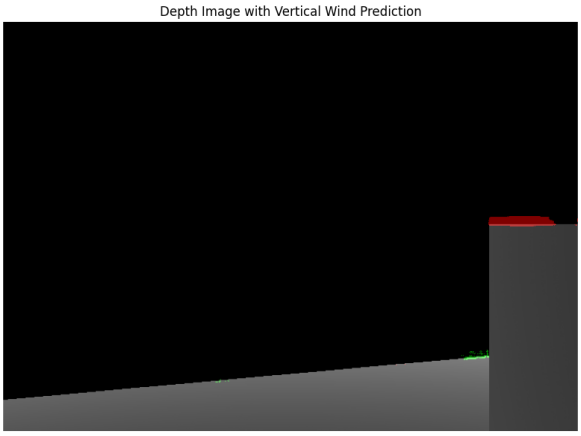


Figure 8.2: Downdraft Prediction, Random Environment

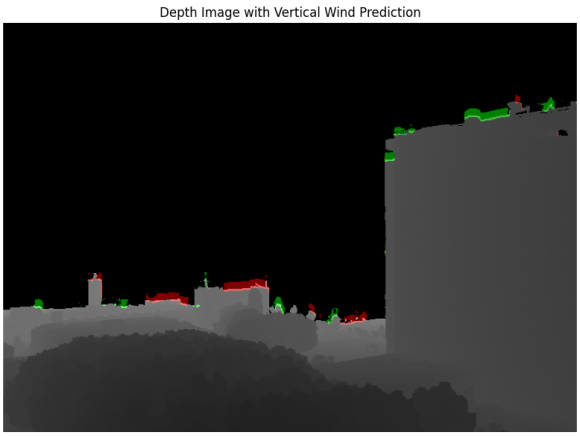


Figure 8.3: Network Prediction 1 TU-Delft environment, windspeed: 5m/s

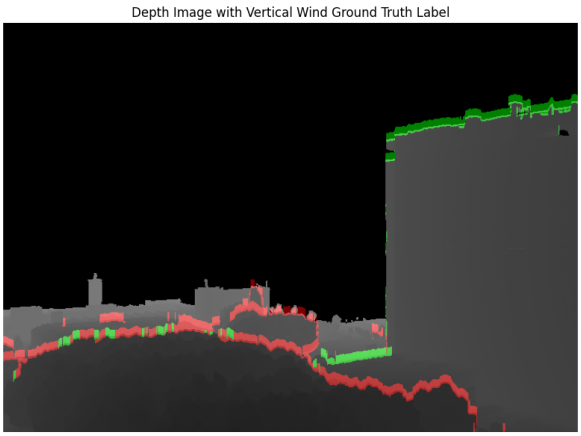


Figure 8.4: Ground Truth Label 1 TU-Delft environment, windspeed: 5m/s

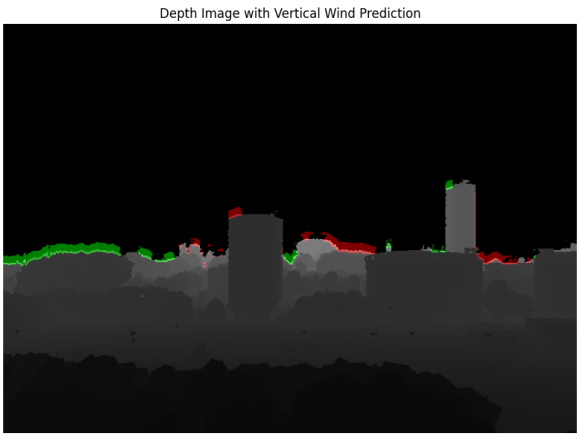


Figure 8.5: Network Prediction 2 TU-Delft environment, windspeed: 5m/s

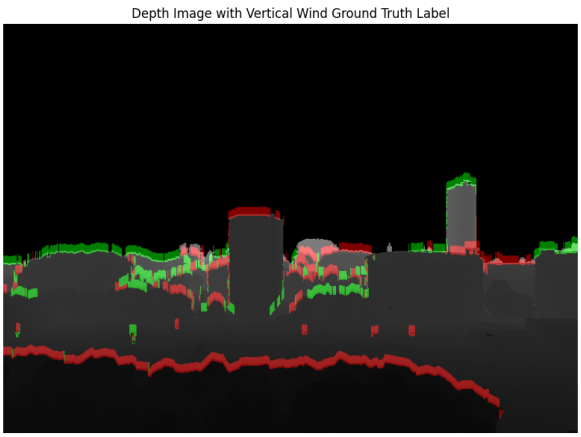


Figure 8.6: Ground Truth Label 2 TU-Delft environment, windspeed: 5m/s

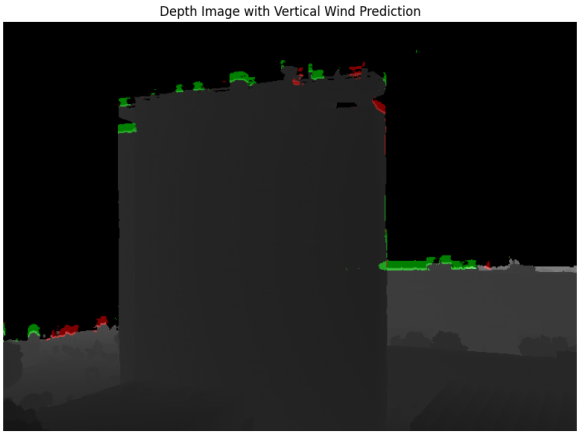


Figure 8.7: Network Prediction 3 TU-Delft environment, windspeed: 5m/s

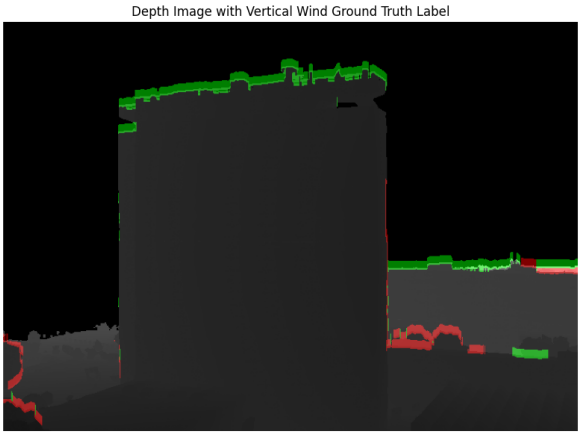


Figure 8.8: Ground Truth Label 3 TU-Delft environment, windspeed: 5m/s

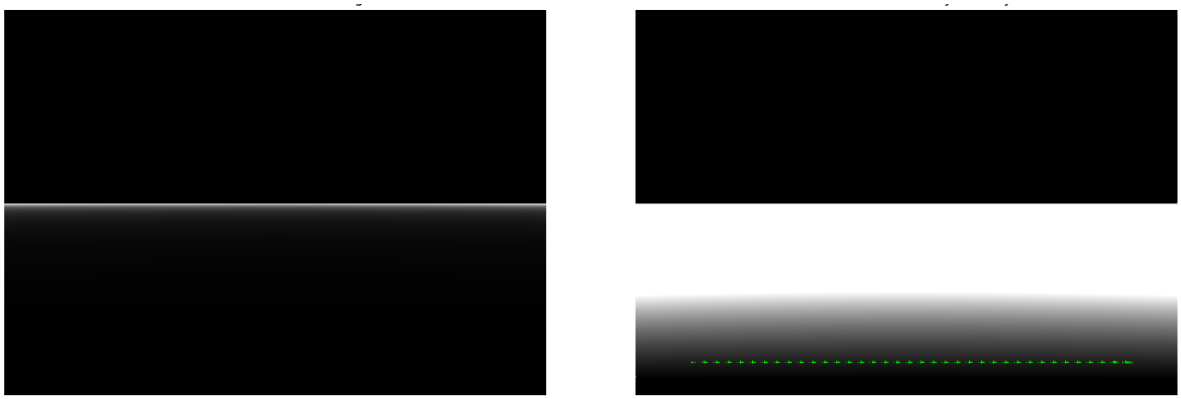


Figure 8.9: Prediction Error due to flat ground surface

Despite these promising results, some misclassifications were noted. Specifically, the model occasionally misidentifies certain features, for example, it sometimes erroneously recognizes the border of the ground plane as a rooftop edge. Such errors indicate that further diversification of the training dataset, particularly by including additional environments with varying ground and rooftop characteristics, could enhance prediction accuracy. Out of approximately 200 images, Figure 8.9 was one of the few predictions that did not make any sense, a result that is generally positive given the computational limits during training.

While Figure 8.10 also produces an incorrect updraft prediction, this error is acceptable since the network might interpret a horizontal flat line as a building ceiling. This issue could potentially be addressed by training on more data or improving the synthetic data labeling process. As discussed in the scientific paper in Part I, this process makes use of edge detection, which likely contributed to the network recognizing the edge in this context.

Now in Figure 8.11 the networks 'horizon' flaw happens again, while still correctly predicting the updraft above the building. Additionally, this observation suggests again that the network might be a bit sensitive to horizon-like features, or more generally horizontal lines. Incorporating more diverse training samples featuring varied horizon lines, that do not have any updraft, and urban edge cases might help solve this issue, leading to more robust predictions overall.

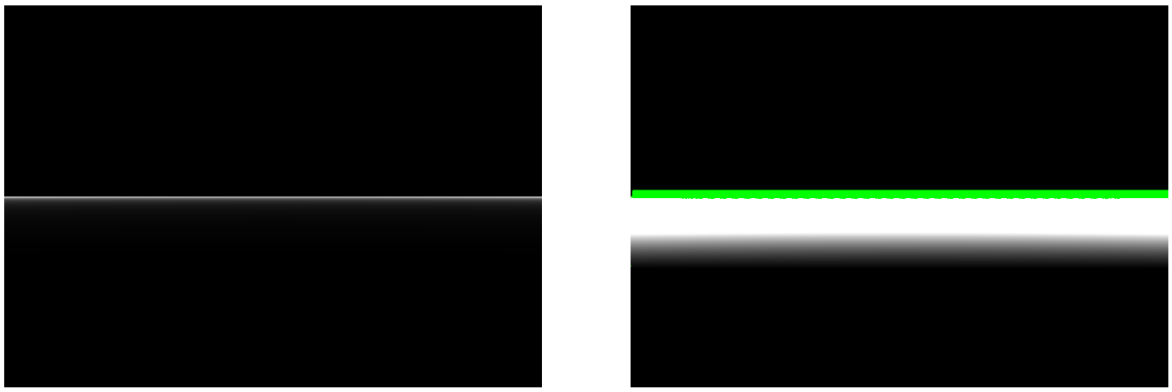


Figure 8.10: Horizon flat surface prediction

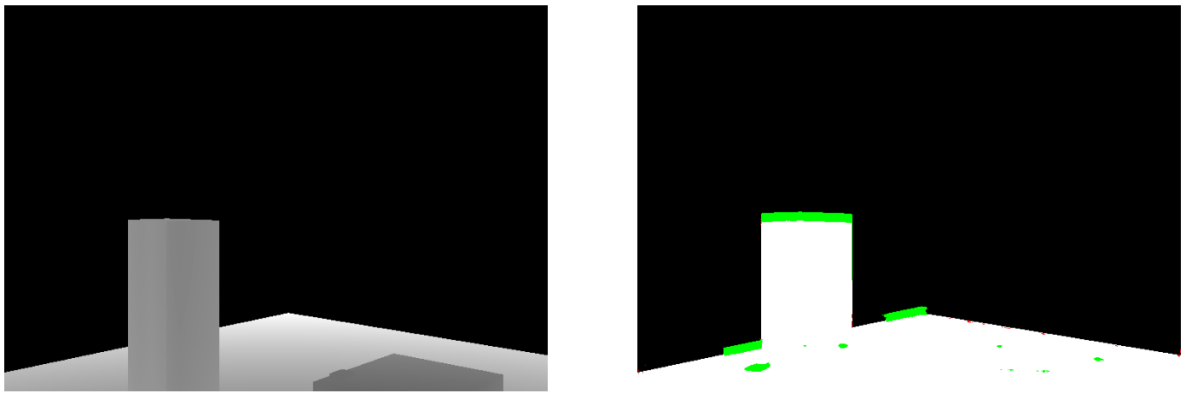
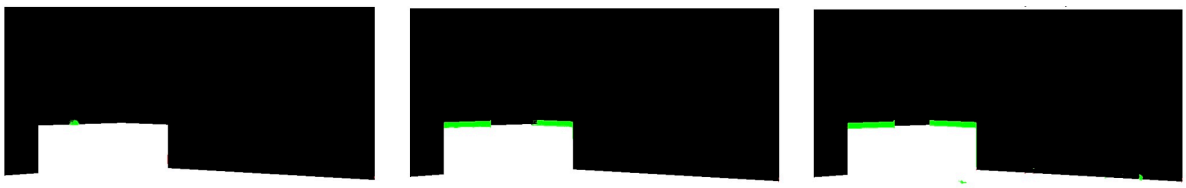


Figure 8.11: Correct updraft prediction, with incorrect horizon network flaw

One observed strength of the model is its adaptability to different wind velocities. At low wind speeds (e.g., 1 m/s), where significant updraft regions are naturally minimal, the model correctly predicts a corresponding lack of updrafts. Conversely, under higher wind velocities, the model effectively identifies more extensive updraft regions, aligning well with expected aerodynamic behavior. This sensitivity to wind speed variations is further evidenced by the accurate predictions of downdraft regions, as discussed before.

Figures Figure 8.12a, Figure 8.12b, and Figure 8.12c illustrate predictions for the same depth image under varying wind velocities. As can be seen, the network shows almost no updrafts at 1 m/s, while the increases to 5 and 10 m/s produce clearly visible changes. This behavior is desirable, as it demonstrates that the model is context-aware and exhibits the appropriate sensitivity to wind speed.



(a) Updraft prediction for 1 m/s wind speed **(b)** Updraft prediction for 5 m/s wind speed **(c)** Updraft prediction for 10 m/s wind speed

Figure 8.12: Updraft predictions for different wind speeds

8.1.1. Suggestions for Future Work and Improvements

To further improve the model's robustness and generalizability, the following enhancements should be considered. These possible improvements are discussed further in chapter 9

- **Quantitative Evaluation Metrics:** Once a more robust model is produced, its important to include additional quantitative metrics such as Mean Absolute Error (Mean Absolute Error (MAE)), Root Mean Squared Error (Root Mean Squared Error (RMSE)), or Intersection over Union (Intersection over Union (IoU)) for segmentation accuracy. These metrics would provide a more detailed evaluation of the model's performance and enable direct comparisons with other approaches.
- **Evaluation on More Environments:** Testing the model on entirely new and diverse urban settings beyond the randomly generated and TU Delft environments. This would help validate the model's generalization capabilities and its adaptability to unforeseen architectural patterns.
- **Edge-Detection Refinement:** Refining the edge-detection component used in pre-labeling by experimenting with advanced techniques or integrating learned edge-detection methods. This may reduce false positives at the ground-to-rooftop transitions, and improve the ground truth label data in general.
- **Real-World Validation:** Supplement synthetic data evaluations with real-world flight tests. This would allow for adjustments based on real aerodynamic conditions and sensor noise, further bridging the gap between simulation and actual drone operation.

By addressing these areas, future iterations of the model would be able to achieve even greater accuracy, which would further increase the usage & usefulness of the model. Once improvements are made this could pave the way to real-life tests.

9 Future Work

In the following, some directions for future work are set, divided into recommendations for the models, improvements to the simulation pipeline, and broader research ideas for which Osprey Simulator could be used.

9.1. Recommendations

To further strengthen and validate the simulation framework, the following is proposed:

- **UAV and Aerodynamics Enhancements:**
  - Develop a fixed-wing UAV model with an articulated USD file that includes control surfaces.
  - Extend the current longitudinal dynamics to full dynamics, incorporating lateral as well as longitudinal behavior.
- **CFD Pipeline Optimization:**
  - Investigate the use of NVIDIA's new ML techniques to enhance CFD simulations [44].
- **Improving Neural Network Predictions (SoarDetect):**
  - Increase the training data size.
  - Increase batch size from 2.
  - Increase the resolution of vertical wind predictions to capture finer aerodynamic details.
- **Real-World Testing & Validation:**
  - Conduct flight experiments with real UAVs to validate simulation predictions.
  - Benchmark SoarDetect's wind predictions against measurements from anemometers or wind LiDAR systems.

9.2. Improvements

Several enhancements can be made to the existing software pipeline, particularly concerning CFD accuracy and its respective computational limits, as well as simulation fidelity:

- **CFD**
  - Fine-tune and optimize CFD parameters to improve simulation accuracy.
- **Computational Optimization:**
  - Optimize CFD simulations through parallelization and GPU-based processing (e.g., CUDA).
  - Design lightweight neural network models for real-time onboard inference.
  - Increase computational capabilities

- **Environment and Computational Enhancements:**
  - Expand the number and variety of real-world environments within the simulator.
  - Increase the environment size and computational limits to support higher-fidelity wind field modeling.
- **Synthetic Data Generation:**
  - Generate more complex urban and natural environments with varied building shapes, materials, and terrain.
  - Improve the synthetic data labeling process

9.3. Research Ideas for Peers

For researchers looking to further this work, the following ideas are proposed:

- **Autonomous Soaring Strategies:**
  - Develop reinforcement learning methods for real-time, energy-optimized path planning towards updrafts.
  - Explore cooperative strategies where multiple UAVs share soaring locations to enhance energy efficiency.
- **Integration with Other Platforms:**
  - Integrate with simulators such as ROS for extended testing and control applications.
  - Explore integration with automatic flight control software like PX4 to enhance overall system performance.

References

[1] Joshua Achiam. “Spinning Up in Deep Reinforcement Learning”. In: (2018).

[2] *AE4010 Research Methodologies: Project Planning*. Aug. 2019.

[3] Zsuzsa Ákos et al. “Thermal soaring flight of birds and unmanned aerial vehicles”. In: *Bioinspiration & biomimetics* 5.4 (2010), p. 045003.

[4] Michael Allen. “Autonomous soaring for improved endurance of a small uninhabitated air vehicle”. In: *43rd AIAA Aerospace Sciences Meeting and Exhibit*. 2005, p. 1025.

[5] Michael J Allen and Victor Lin. *Guidance and control of an autonomous soaring UAV*. Tech. rep. 2007.

[6] Ahmad Taher Azar et al. “Drone Deep Reinforcement Learning: A Review”. In: *Electronics* 10.9 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10090999. URL: https://www.mdpi.com/2079-9292/10/9/999.

[7] Randall Balestriero et al. *A Cookbook of Self-Supervised Learning*. 2023. arXiv: 2304.12210 [cs.LG].

[8] Blender. *Blender - Simulations*. Accessed: 26 August 2023. URL: %5Curl%7Bhttps://www.blender.org/features/simulation/#fluids%7D.

[9] Anjan Chakrabarty and Jack Langelaan. “Flight path planning for uav atmospheric energy harvesting using heuristic search”. In: *AIAA guidance, navigation, and control conference*. 2010, p. 8033.

[10] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: 1604.01685 [cs.CV].

[11] NVIDIA Corporation. *Isaac Sim Documentation*. https://docs.omniverse.nvidia.com/isaacsim/latest/index.html. Accessed: 2024-12-03. 2025.

[12] NVIDIA Corporation. *OpenUSD on NVIDIA Omniverse*. https://www.nvidia.com/en-us/omniverse/usd/?ncid=pa-srch-goog-230293-vt49&_bt=697115562887&_bk=openusd&_bm=p&_bn=g&_bg=162278196418&gad_source=1&gclid=EAIaIQobChMI3YzrqYG7hgMV2zWtBh2QhwopEAAYASAAEgJbVfD_BwE. Accessed: 2024-12-03. 2025.

[13] Tim De Craecker. *Creating a City LandScape in Blender 3D*. https://medium.com/@timdecraecker/creating-a-city-landscape-in-blender-3d-a06015eb0a1b. Accessed: 25 August 2023. 31 May 2021.

[14] Tim De Craecker. *Creating Urban Landscape in Blender 3D — Part 2*. Accessed: 25 August 2023. 31 May 2021. URL: %5Curl%7Bhttps://medium.com/@timdecraecker/creating-urban-landscape-in-blender-3d-part-2-a5880ec8bf8e%7D.

[15] Nvidia Developer. *Isaac Gym*. Accessed: 28 August 2023. URL: %5Curl%7Bhttps://developer.nvidia.com/isaac-gym%7D.

[16] Dharmaraj. *Convolutional Neural Networks (CNN) - Architecture Explained*. https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243. Accessed: 15 September 2023. 1June 2023.

[17] Daniel Elfverson and Christian Lejon. “Use and Scalability of OpenFOAM for Wind Fields and Pollution Dispersion with Building- and Ground-Resolving Topography”. In: *Atmosphere* 12.9 (2021). ISSN: 2073-4433. DOI: 10.3390/atmos12091124. URL: https://www.mdpi.com/2073-4433/12/9/1124.

[18] I-Sheng Fang et al. “ES3Net: Accurate and Efficient Edge-Based Self-Supervised Stereo Matching Network”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4472–4481.

[19] Alex Fisher et al. “Micro air vehicle soaring in urban environments”. In: *2016 Australian Control Conference (AuCC)*. IEEE. 2016, pp. 9–14. DOI: 10.1109/AUCC.2016.7867924.

[20] Philipp Foehn et al. “Alphapilot: Autonomous drone racing”. In: *Autonomous Robots* 46.1 (2022), pp. 307–320.

[21] OpenFOAM Foundation. *Chapter 1 Introduction*. Accessed: 26 August 2023. URL: %5Curl%7Bhttps://www.openfoam.com/documentation/user-guide/1-introduction%7D.

[22] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[24] Vitor Guizilini et al. *3D Packing for Self-Supervised Monocular Depth Estimation*. 2020. arXiv: 1905.02693 [cs.CV].

[25] Sunyou Hwang, Bart DW Remes, and Guido CHE de Croon. “AOSoar: Autonomous Orographic Soaring of a Micro Air Vehicle”. In: *arXiv preprint arXiv:2308.00565* (2023).

[26] Marcelo Jacinto et al. *Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation*. 2023. arXiv: 2307.05263 [cs.RO].

[27] Tor A. Johansen and Thor I. Fossen. “Control allocation—A survey”. In: *Automatica* 49.5 (2013), pp. 1087–1103. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.01.035. URL: https://www.sciencedirect.com/science/article/pii/S0005109813000368.

[28] Chris PL de Jong et al. “Never landing drone: Autonomous soaring of a unmanned aerial vehicle in front of a moving obstacle”. In: *International Journal of Micro Air Vehicles* 13 (2021), pp. 1–12.

[29] Alexander Kirillov et al. “Panoptic Feature Pyramid Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[30] Alexander Kirillov et al. *PointRend: Image Segmentation as Rendering*. 2020. arXiv: 1912.08193 [cs.CV].

[31] Alexandros Kouris and Christos-Savvas Bouganis. “Learning to Fly by MySelf: A Self-Supervised CNN-Based Approach for Autonomous Navigation”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594204.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[33] Mihir Kulkarni, Theodor J. L. Forgaard, and Kostas Alexis. *Aerial Gym – Isaac Gym Simulator for Aerial Robots*. 2023. arXiv: 2305.16510 [cs.RO].

[34] Inwoong Lee et al. “An Efficient Human Instance-Guided Framework for Video Action Recognition”. In: *Sensors* 21 (Dec. 2021), p. 8309. DOI: 10.3390/s21248309.

[35] Shushuai Li, Christophe De Wagter, and Guido CHE De Croon. “Self-supervised monocular multi-robot relative localization with efficient deep neural networks”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 9689–9695. DOI: 10.1109/ICRA46639.2022.9812150.

[36] Yen-Cheng Liu et al. *Unbiased Teacher for Semi-Supervised Object Detection*. 2021. arXiv: 2102.09480 [cs.CV].

[37] Emmanuel Maggiori et al. “Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark”. In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE. 2017.

[38] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021. arXiv: 2108.10470.

[39] Maximilian Menke, Thomas Wenzel, and Andreas Schwung. “Improving Cross-Domain Semi-Supervised Object Detection with Adversarial Domain Adaptation”. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–7. DOI: 10.1109/IV55152.2023.10186678.

[40] Abdulghani Mohamed et al. “Opportunistic soaring by birds suggests new opportunities for atmospheric energy harvesting by flying robots”. In: *Journal of the Royal Society, Interface* 19 (Nov. 2022), p. 20220671. DOI: 10.1098/rsif.2022.0671.

[41] Abdulghani Mohamed et al. “Scale-resolving simulation to predict the updraught regions over buildings for MAV orographic lift soaring”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 140 (2015), pp. 34–48.

[42] Joshua C Nathanael, Chung Hung J Wang, and Kin Huat Low. “Simulation of Wind Field in a Building Complex for Evaluation of the Wind Effect Along UAS Flight Path”. In: *AIAA AVIATION 2023 Forum*. 2023, p. 4096.

[43] Stefan Notter et al. “Hierarchical Reinforcement Learning Approach for Autonomous Cross-Country Soaring”. In: *Journal of Guidance, Control, and Dynamics* 46.1 (2023), pp. 114–126.

[44] NVIDIA. *Transforming CFD Simulations with ML using NVIDIA Modulus*. https://developer.nvidia.com/blog/transforming-cfd-simulations-with-ml-using-nvidia-modulus/. Accessed: 2024-12-12. 2020.

[45] Matthew Penn et al. “A method for continuous study of soaring and windhovering birds”. In: *Scientific Reports* 12.1 (2022), p. 7038.

[46] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. “SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9250–9256. DOI: 10.1109/ICRA.2019.8793621.

[47] Lord Rayleigh. “The soaring of birds”. In: *Nature* 27.701 (1883), pp. 534–535.

[48] Gautam Reddy et al. “Glider soaring via reinforcement learning in the field”. In: *Nature* 562.7726 (2018), pp. 236–239.

[49] Philip L Richardson. “Upwind dynamic soaring of albatrosses and UAVs”. In: *Progress in Oceanography* 130 (2015), pp. 146–156.

[50] RIGITECH. *Technology*. Accessed: 16 August 2023. URL: %5Curl%7Bhttps://rigi.tech/technology/%7D.

[51] Skyports Drone Services. *Royal Mail and Skyports partner on drone delivery*. https://skyportsdroneservices.com/royal-mail-and-skyports-partner-on-drone-delivery/. Accessed: 09 August 2023. 16 December 2020.

[52] Shital Shah et al. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *Field and Service Robotics*. 2017. eprint: arXiv:1705.05065. URL: https://arxiv.org/abs/1705.05065.

[53] V Kr Sharan. “On characteristics of flow around building models with a view to simulating the minimum fraction of the natural boundary layer”. In: *International journal of mechanical sciences* 17.9 (1975), pp. 557–563.

[54] Ana-Maria Simion and □erban Radu. “Experiments with Semi-Supervised Learning: from Cityscapes to Medical Images”. In: *2023 24th International Conference on Control Systems and Computer Science (CSCS)*. 2023, pp. 477–483. DOI: 10.1109/CSCS59211.2023.00081.

[55] Chahat Deep Singh et al. “WorldGen: A Large Scale Generative Simulator”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9147–9154.

[56] Yunlong Song et al. “Flightmare: A flexible quadrotor simulator”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 1147–1157.

[57] Ian A Stokes and Andrew J Lucas. “Wave-slope soaring of the brown pelican”. In: *Movement Ecology* 9 (2021), pp. 1–13.

[58] Tom Suys et al. “Autonomous Control for Orographic Soaring of Fixed-Wing UAVs”. In: *arXiv preprint arXiv:2305.13891* (2023).

[59] Tanmaya Swain et al. “Deep Reinforcement Learning based Target Detection for Unmanned Aerial Vehicle”. In: *2022 IEEE India Council International Subsections Conference (INDISCON)*. 2022, pp. 1–5. DOI: 10.1109/INDISCON54605.2022.9862891.

[60] Jake Tallman. “Soarnet, Deep Learning Thermal Detection for Free Flight”. PhD thesis. California Polytechnic State University, 2021.

[61] Henk Tennekes. *The Simple Science of Flight, Revised and Expanded Edition: From Insects to Jumbo Jets*. MIT press, 2009.

[62] JJ Videler, D Weihs, and S Daan. “Intermittent gliding in the hunting flight of the kestrel, Falco tinnunculus L”. In: *Journal of experimental Biology* 102.1 (1983), pp. 1–12.

[63] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[64] Simon Watkins et al. “Towards autonomous MAV soaring in cities: CFD simulation, EFD measurement and flight trials”. In: *International Journal of Micro Air Vehicles* 7.4 (2015), pp. 441–448.

[65] Caleb White et al. “A feasibility study of micro air vehicles soaring tall buildings”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 103 (2012), pp. 41–49.

[66] Caleb White et al. “The soaring potential of a micro air vehicle in an urban environment”. In: *International Journal of Micro Air Vehicles* 4.1 (2012), pp. 1–13.

[67] Wikipedia contributors. *Thermal — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Thermal&oldid=1125892184. [Online; accessed 26-September-2023]. 2022.

[68] Timothy D Woodbury, Caroline Dunn, and John Valasek. “Autonomous soaring using reinforcement learning for trajectory generation”. In: *52nd Aerospace Sciences Meeting*. 2014, p. 0990.

[69] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[70] Yumin Tan Wuttichai Boonpook and Bo Xu. “Deep learning-based multi-feature semantic segmentation in building extraction from images of UAV photogrammetry”. In: *International Journal of Remote Sensing* 42.1 (2021), pp. 1–19. DOI: 10.1080/01431161.2020.1788742. URL: %5Curl%7Bhttps://doi.org/10.1080/01431161.2020.1788742%7D.

[71] Ishan Misra Yann LeCun. *Self-supervised learning: The dark matter of intelligence*. https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/. Accessed: 22 August 2023. 4March 2021.

[72] Hang Yu, Guido C. H. E de Croon, and Christophe De Wagter. *AvoidBench: A high-fidelity vision-based obstacle avoidance benchmarking suite for multi-rotors*. 2023. arXiv: 2301.07430 [cs.RO].

[73] Zhiding Yu et al. “Casenet: Deep category-aware semantic edge detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5964–5973.

A

Supplementary Material

Source Code Osprey Simulator: `https://github.com/kv8-A/OspreySimulator`