Westfälische Hochschule

Scientific specialization

ROSHAN: Rescue Oriented Simulation: Handling and Navigating Fires

Julien Meine

January 25, 2024

Supervising professor
Prof. Dr.-Ing. Dipl. Inform. Hartmut Surmann

Department
Information technology and communication

Abstract

This paper addresses the escalating challenges in wildfire management, underscored by increasing frequencies and intensities of wildfires globally. It introduces ROSHAN (Rescue Oriented Simulation: Handling and Navigating Fires), a wildfire simulation tool. ROSHAN integrates the principles of cellular automata with reinforcement learning to simulate wildfire dynamics and automatic handling. The paper delves into the intricacies of wildfire modeling, emphasizing the implementation and functionalities of ROSHAN. Central to this simulation tool is its rendering and simulation of fire spread, achieved by integrating data from the CORINE database. The interactive graphical interface of ROSHAN facilitates real-time monitoring and manipulation of fire scenarios. A key component in ROSHAN is the incorporation of a Reinforcement Learning agent, embodied as a drone, which learns to detect and mitigate fires. While acknowledging the current limitations of this tool, the paper also explores potential future enhancements.

Keywords: Simulation; Cellular Automata; Fire modelling; Reinforcement Learning

Contents

Introduction

In recent years, the escalating frequency and intensity of vegetation fires have emerged as a significant concern, posing risks not only to natural ecosystems but also challenging existing paradigms in fire management and control [1] [2]. This escalation, attributed to a confluence of factors including climate variability, land use changes, and urban expansion, has necessitated innovative approaches in understanding and mitigating fire spread.

Computational simulations have increasingly become indispensable in the study of wildfire behavior, offering controlled environments to test various hypotheses and scenarios.

> "In recent years, research on wildfires has primarily focused on wildfire spread behavior, wildfire models [3], wildfire simulation and spatiotemporal distribution [4], wildfire monitoring [5], wildfire hazard risk assessment [6], wildfire impacts on the ecological environment [7], and post-fire ecological restoration. This indicates that understanding and studying the spread of wildfires has been a persistent research focus." (Sun et al., 2023) [8]

These simulations are pivotal in deciphering the complex dynamics of fire spread, enabling researchers to explore and validate strategies for effective fire management. The importance of such simulations is magnified when considering the unpredictable nature of wildfires, where traditional methods often fall short in providing real-time, actionable insights.

The current work is anchored in this realm of simulation, focusing on the development and refinement of a tool - the ROSHAN simulation. This tool, designed for the purpose of research, stands at the forefront of this study. ROSHAN, an acronym for "Rescue Oriented Simulation: Handling and Navigating Fires", utilizes fire spread models to predict and visualize fire spread. It aims to offer a robust platform for studying fire spread dynamics and training reinforcement learning agents.

ROSHAN's core simulation employs a cellular automaton framework, a method that has demonstrated efficacy in modeling complex, dynamic systems such as wildfires. This approach facilitates a granular understanding of fire spread, accounting for varied parameters like vegetation type, topography, and climatic conditions. The inclusion of a Reinforcement Learning agent, in the form of an autonomous drone, is a key feature of the ROSHAN simulation, marking a interesting contribution to wildfire management.

This agent navigates and interacts with simulated fire scenarios, offering practical data that could be vital for improving emergency response tactics.

While the focus of this paper is not primarily on the Wildland-Urban Interface, the lessons learned and methodologies developed herein are broadly applicable, including in WUI contexts. The primary goal is to create a simulation tool focused on accurately modeling wildfires in a range of locations, including but not limited to Wildland-Urban Interface areas.

This paper intentionally omits a dedicated 'Related Work' chapter. This decision stems from the extensive exploration of Cellular Automata (CA) approaches for wildfire simulation previously conducted in earlier work. Readers seeking detailed insights into the various CA-based wildfire simulation methodologies and their comparative analyses are referred to "Modellierung von Bränden mit zellulären Automaten" [9], where a comprehensive review of related studies in this domain is presented. The current paper focuses more on advancing the practical application and simulation aspects, building upon the foundational understanding established in the prior work.

Foundational Methods

Wildfire Models

In this chapter, a diverse range of models employed in wildfire simulation are discussed, each contributing uniquely to the understanding and management of wildfire dynamics. These models are generally categorized into three types: Empirical, Semi-Empirical, and Physically Based Models, each offering different perspectives and tools for predicting and analyzing fire behavior [10] [11].

Empirical Models

Empirical models are grounded in statistical analysis and historical fire data. They are primarily used for predicting fire behavior based on established patterns and observed trends from past fires. Their strength lies in their simplicity and ease of use, making them highly effective for rapid predictions in various scenarios, particularly during the initial stages of fire response. However, their reliance on historical data may limit their accuracy in novel or highly variable conditions. Notable examples of empirical models

include McArthur's model [12] for grassland and forest fires in Australia and the Canadian Forest Fire Behavior Prediction System [13], which integrates decades of research into a comprehensive prediction tool. These models are essentially statistical descriptions of fire spread, relying soley on historical patterns and pay no intention on physical fire mechanisms.

Semi-Empirical Models

Semi-Empirical models strike a balance between empirical observations and physical modeling principles. These models aim to blend the accuracy of physical models with the applicability of empirical models, offering a pragmatic approach for operational fire management. Rothermel's model [15], for instance, is a widely-used semi-empirical model in the United States, providing equations for calculating the rate of fire spread and fire intensity. A well known software that incorporates Rothermel's model, and many other such models, is FARSITE [16], which applies semi-empirical relationships to calculate fire spread along the surface and includes considerations for ground-to-crown transitions(see Figure 1). The Prometheus Fire Simulator [17] is a sophisticated model designed for predicting and analyzing the spread of wildfires, incorporating the Canadian Forest Fire Behavior Prediction System [13] and Huygens' principle of wave propagation to simulate fire behavior in different landscapes. These models are less computationally intensive than purely physical models but can still require substantial computational resources and empirical data for calibration or validation.

Physically Based Models

Physically based models are the most complex, utilizing the fundamental laws of physics and chemistry to simulate fire behavior. They offer high-fidelity simulations, often providing detailed and accurate representations of fire dynamics. These models incorporate aspects like radiation as the dominant heat transfer mechanism and the effects of wind and slope on fire spread. Examples include NCAR's Coupled Atmosphere-Wildland Fire-Environment model [18], or the Wildland Fire Dynamics Simulator [19], which integrate computational fluid dynamics with wildfire components. While they offer a high degree of precision, their complexity results in substantial computational demands and necessitates extensive data on fuel composition, terrain, and weather conditions. These models are particularly useful for in-depth research and planning large-scale fire management strategies.

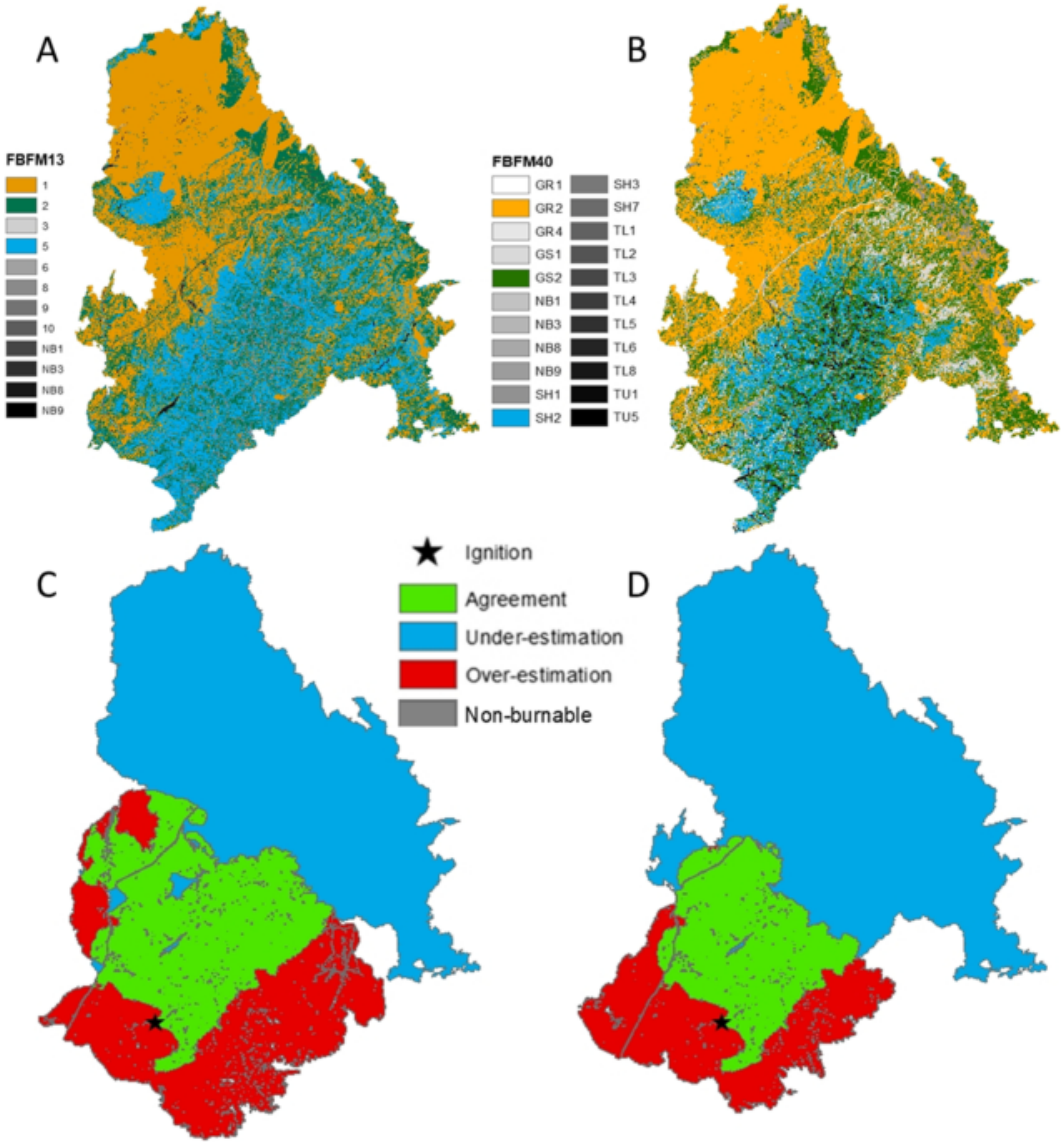

Figure 1: Overview of the FARSITE Simulation of a fire occuring in the sagebrush steppe in the western USA. Modelling and research done by Price et al. [14]. Maps A and B show different inputs of Fire Behaviour Fuel Models, both used as an input from FARSITE. Maps C and D show the respective output of the Simulation. The blue area shows where the real fire burned down the vegetation, but the FARSITE simulation failed to simulate. The sum of the red and green area is FARSITE's simulated burned down area.

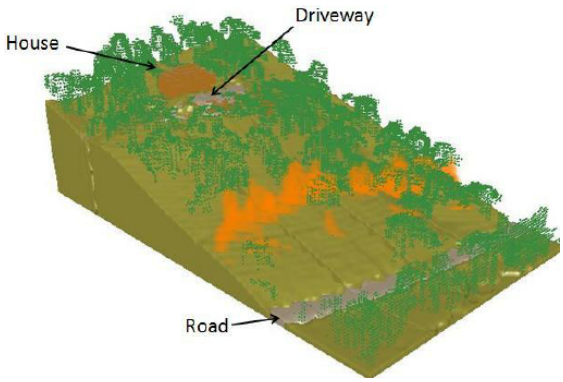

Figure 2: Example of a Wildland Fire Dynamics Simulator simulation, visualized using Smokeview, showcasing a fire near a house. The simulation results were produced by Mell et al. [20]

Limitations

Understanding the diverse range of fire simulation models is crucial for selecting the appropriate approach for specific fire management objectives. While physical models provide detailed simulations, their computational demands limit their widespread applicability. Semi-empirical models offer a balance, making them suitable for operational use, whereas empirical models, with their simplicity and rapid output, are ideal for initial response scenarios. This overview highlights the importance of aligning the model choice with specific fire management needs and constraints.

Cellular Automata

Cellular Automata, a concept rooted in the realm of mathematics and computer science, represent a compelling and robust approach to modeling complex systems through simple components. At its core, a Cellular Automaton consists of a grid of cells, each of which can be in one of a finite number of states. The grid can be of any dimension, but most commonly, CA are implemented in two dimensions for practical and visual clarity.

The evolution of a Cellular Automaton is governed by a set of rules, which are applied at discrete time steps and determine the state of each cell based on the states of its neighbors. These neighbors are typically the adjacent cells in the grid, and the interaction rule applied is uniform across the grid. What makes CA particularly fascinating is the

emergence of complex, often unexpected behaviors from these simple, local interactions. Despite the deterministic nature of the rules, the global behavior of the system can exhibit rich, dynamic, and sometimes chaotic patterns [21].

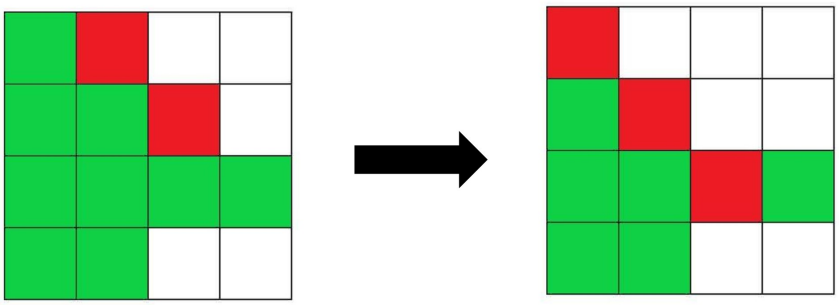

Figure 3: Illustration of the Drossel-Schwabl [22] forest-fire model in one simulation step. In this model, cells follow four fundamental rules: (1) A tree grows in an empty cell with probability p, (2) A tree ignites with probability f, (3) A tree burns if adjacent to a burning tree, utilizing the von Neumann neighborhood for adjacency, and (4) A burning cell becomes empty. The depicted states are: burning trees (red cells), occupied by trees (green cells), and empty (white cells).

In the context of fire simulation, as explored in "Modellierung von Bränden mit zellulären Automaten" [9], Cellular Automata can be good option in modelling firespread. Fire, as a natural process, exhibits complex behavior that emerges from relatively simple and local interactions. The spread of fire across a landscape is influenced by the properties of the local vegetation, the topography, and weather conditions, all of which can be effectively modeled using the CA framework. Each cell in the CA grid can represent a patch of land, with its state indicating whether it is burning, has burnt out, or is yet to catch fire. The rules governing the state transitions can encapsulate factors such as the likelihood of a cell igniting based on the heat from neighboring cells, the presence and type of fuel, and wind direction and speed.

This approach to modeling fire using Cellular Automata aligns well with the principles of simplicity and local interaction that CA embodies. By adjusting the rules and parameters of the CA model, a wide range of fire behaviors can be simulated, from slow, smoldering fires in wet, dense vegetation to rapid crown fires in dry, windy conditions.

The choice to employ Cellular Automata for fire simulation in the ROSHAN project is anchored in a strategic balance between computational efficiency and the potential for future development. Compared to other fire simulation methodologies, which can be

computationally intensive, CA offers a more streamlined approach. This efficiency is not merely a matter of convenience; it is a critical factor in real-time applications and, more importantly, in the realm of Reinforcement Learning.

Training an Reinforcement Learning agent demands a simulation environment that can provide rapid feedback and adapt to the evolving strategies of the agent. In this context, more precise, yet computationally demanding simulations, such as the Fire Dynamics Simulator, though offering detailed accuracy, become less feasible. These models, while thorough in their representation of fire dynamics, require significant computational resources, slowing down the iterative learning process essential in Reinforcement Learning.

CA, by contrast, provides a swift and effective means of simulating fire spread. This rapid modeling capability has been recognized and utilized in various advanced simulations, where CA methods are often integrated with other modeling techniques to balance detail and speed [23] [24] [25] [26]. The inherent simplicity and speed of CA make it an ideal primary approach for the initial stages of the ROSHAN project, where the primary objective is to develop and train an efficient Reinforcement Learning agent.

Furthermore, the decision to use CA is also influenced by a long-term view of the project's evolution. The modular nature of CA, along with its relative simplicity, opens up possibilities for future enhancements. There is potential for integrating CA with more complex models, or even developing hybrid systems that combine the strengths of various simulation approaches. This flexibility aligns with the goal of continually refining the simulation, enhancing its accuracy and applicability in fire management and emergency response scenarios.

In summary, the utilization of Cellular Automata in the ROSHAN simulation is a strategic choice that optimizes for current Reinforcement Learning training needs while keeping doors open for future advancements in fire simulation technology.

Brief Introduction to Reinforcement Learning

Reinforcement Learning is a unique domain within the broader field of machine learning, characterized by its approach to learning through interaction and feedback. Unlike supervised learning, which relies on labeled datasets for training models, or unsupervised learning, which seeks to find patterns in unlabeled data, Reinforcement Learning is centered around the concept of an agent learning from the consequences of its actions in a dynamic environment. This distinction is fundamental, as Reinforcement Learning

does not require a predefined dataset but instead gathers knowledge through trial and error, adapting its strategies based on received feedback.

In Reinforcement Learning, the learning process is not directed by explicit instructions. Instead, it is driven by the agent's experiences as it navigates through a problem space. This paradigm shift from a data-centric approach (as seen in supervised and unsupervised learning) to an experience-based approach positions Reinforcement Learning in a unique spot where it can tackle problems involving sequential decision-making and uncertainty - areas where traditional machine learning approaches might struggle.

Basic Principles

The core of reinforcement learning is an iterative process involving actions, feedback, and rewards. An Reinforcement Learning agent interacts with its environment by performing actions, to which the environment responds with new states and feedback in the form of rewards or penalties. The agent's objective is to learn a strategy, or policy, that maximizes the cumulative reward over time.

This process can be viewed as a continuous loop where the agent:

- Observes the current state(S_t) of the environment.
- Selects and performs an action(A_t) based on its current policy.
- Receives feedback through rewards(R_t) or penalties.
- Adjusts its policy to improve future rewards.

The reward system is crucial in Reinforcement Learning. It essentially encodes the goals the agent is meant to achieve. A reward is a scalar feedback signal that tells the agent how well it is performing at a given step. The agent's job is to maximize the total amount of reward it receives over the long run. This involves not just immediate rewards but also the strategic consideration of long-term consequences of actions.

In summary, the fundamental principle of Reinforcement Learning is learning through interaction. The agent's ability to evaluate its actions based on trial-and-error experiences and adapt its policy towards actions that yield the highest cumulative rewards is what makes Reinforcement Learning a powerful tool for solving complex, dynamic problems that require a sequence of decisions [27].

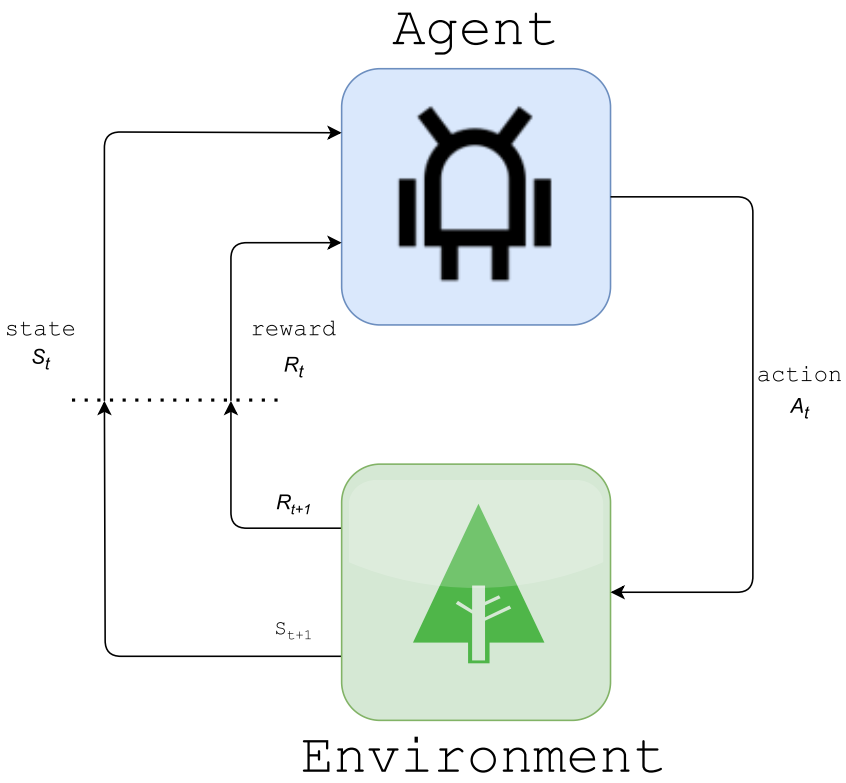

Figure 4: Reinforcement Learning Paradigm

Core Components of Reinforcement Learning

Reinforcement Learning is characterized by several core components that define its functionality and applicability. These components interact with each other to create a dynamic learning environment where an agent learns to perform tasks effectively.

Agent and Environment Dynamics

In Reinforcement Learning, the dynamics between the agent and the environment form the cornerstone of the learning process. An agent, serving as the decision-maker, interacts with an environment that encapsulates the specific problem or task at hand. This interaction is inherently cyclical and dynamic. It begins with the agent observing the current state of the environment, which acts as a cue for decision-making. Based on this observation, the agent executes an action, influencing the environment. In response, the environment transitions to a new state and presents feedback to the agent in the form

of a reward. This feedback loop is continuous, fostering an ongoing learning process where the agent constantly refines its actions to achieve better outcomes.

States, Actions, and Rewards

Central to this interactive learning loop are the concepts of states, actions, and rewards. A state represents a specific situation or configuration in the environment, offering the agent the necessary context for decision-making. The set of all possible states forms the environment's landscape within which the agent operates. Actions, then, are the decisions or moves the agent makes in response to the state it observes. Each action taken by the agent alters the state of the environment, potentially leading to different outcomes. The environment evaluates these actions and conveys this through rewards - numerical values that signify the benefit or cost associated with the agent's actions. The agent's primary objective is to maximize its cumulative rewards over time, guiding its learning trajectory towards more effective strategies.

Policies

A policy in Reinforcement Learning is essentially the agent's strategy for action selection in various states. It's a rule or set of rules that dictates how the agent responds to different states in the environment. Policies can be deterministic, where a specific action is consistently chosen for each state, or stochastic, where actions are chosen based on a probability distribution. The effectiveness of a policy is measured by the amount of cumulative reward it can garner. The learning journey of an Reinforcement Learning agent is fundamentally about discovering or refining policies that maximize these rewards, thereby achieving the desired goals more efficiently.

Value Functions

Integral to the Reinforcement Learning framework are the value functions, which are instrumental in assessing the potential of states or actions. There are two primary types: the state-value function and the action-value function. The state-value function offers an expectation of the cumulative reward that can be obtained from a particular state, assuming the agent follows a specific policy thereafter. On the other hand, the action-value function predicts the expected cumulative reward for choosing a specific action in a given state, followed by adherence to a policy. These value functions are important in guiding the agent's policy selection, enabling it to estimate the long-term benefits of its actions.

Environment Models

The role of environment models in Reinforcement Learning varies depending on the approach. In model-based Reinforcement Learning, an explicit model of the environment

is used to anticipate the outcomes of actions, including the next state and the expected reward. This approach allows for planning and foresight in decision-making. Conversely, in model-free Reinforcement Learning, the agent learns directly from its interactions with the environment without relying on a predefined model. It learns the value functions or the optimal policy through accumulated experiences. Regardless of the approach, understanding the environment, be it through direct interaction or modeling, is crucial as it fundamentally shapes the agent's learning process and the development of effective policies.

Overview of Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an algorithm in reinforcement learning that belongs to the family of policy gradient methods. PPO aims to optimize the policy directly, ensuring effective learning while maintaining computational efficiency. Its core objective is to update policies in a way that maximizes the expected return while minimizing the deviation from the previous policy [28].

Key Features of PPO Objective Function and Clipped Surrogate Objective

The objective function of PPO, central to its operation, is designed to balance the need for new policy exploration while avoiding significant deviations from the current policy. This balance is achieved through a concept called the clipped surrogate objective. The mathematical formulation of this objective is:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t) \right] \tag{0.1}$$

where:
θ represents the policy parameters.
$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio of the action under the new policy π_θ to the old policy $\pi_{\theta_{old}}$.
$\hat{A}_t$ is an estimator of the advantage function at time t.
ε is a hyperparameter that defines the clipping range.

The clipping mechanism in PPO's objective function limits the update extent, ensuring that the new policy remains 'proximal' to the old policy. This prevents large, potentially destabilizing updates to the policy.

Advantage Function Estimation

PPO relies on estimating the advantage function $\hat{A}_t$, which indicates how much better or worse an action is compared to the policy's average. This estimation is crucial for determining the direction and magnitude of the policy update.

Stable Policy Updates

One of the key strengths of PPO is its ability to achieve stable policy updates. The clipped surrogate objective ensures that the policy does not change too abruptly, which is a common challenge in Reinforcement Learning. This stability is vital for the learning process, particularly in environments with high variance or complexity.

Balancing Exploration and Exploitation

PPO efficiently balances exploration (trying new actions) and exploitation (using known high-reward actions). This balance is achieved through the adaptive nature of the clipping mechanism in the objective function, which restricts the policy updates to a reasonable range. By doing so, PPO ensures consistent learning progress and avoids the pitfalls of excessive exploration or premature convergence.

ROSHAN

Overview

ROSHAN, an acronym for ”Rescue Oriented Simulation: Handling and Navigating Fires”, is a program designed for wildfire simulation. The simulation is rooted in the concept of cellular automata and features a graphical interface that allows real-time observation of the fire front. The interface not only provides a dynamic visual representation of the fire spread but also enables users to make various inputs. These range from the initial setup of the simulation, such as loading the map and setting the initial parameters, to real-time control of the simulation process.

For instance, users have the ability to set fire starting points by clicking on individual cells, initiating the spread of the fire front from these points. Conversely, clicking on an already burning cell extinguishes the fire there. This interactive feature provides a hands-on experience, making it easier to understand and analyze the fire’s behavior. Additionally, the interface allows users to query information about the current state of each cell, enhancing the level of detail and control available during the simulation.

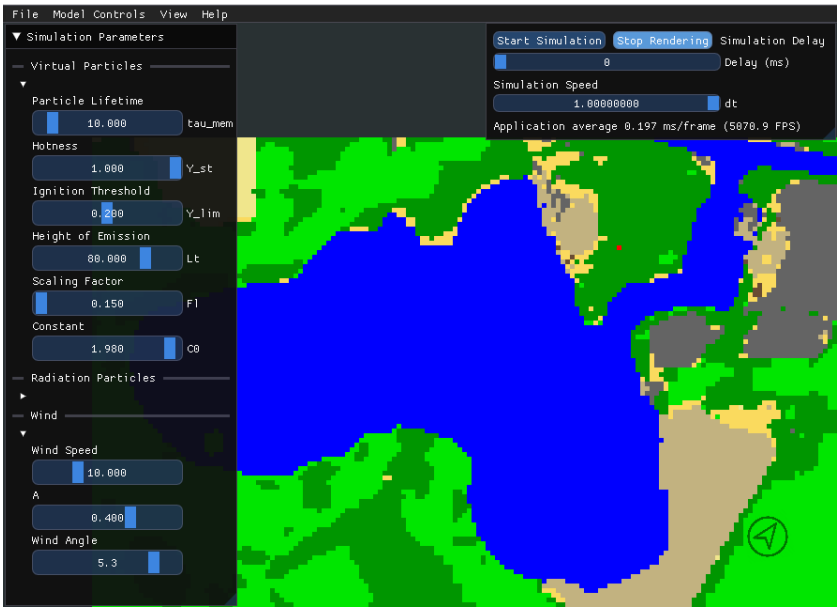

Figure 5: Overview of the ROSHAN simulation. The loaded map is an artificial lake located in Haltern am See at ”Die Haard”.

The fire model in ROSHAN is based on the generation of virtual particles(see Figure 6) that can ignite cells. These particles are also displayed on the interface, providing a comprehensive view of how fire spreads across the landscape. This feature not only adds to the visual realism of the simulation but also aids in understanding the underlying mechanics of fire propagation in the model.

Figure 6: A small area on fire. Dots in the image represent particles. The size and ”yellowness” of the particle determines it’s ”hotness”. Small, red particles existed for longer timesteps and will extinguish soon.

A significant aspect of ROSHAN is its Python bindings, which enable users to run the simulation and query various parameters through Python. This integration has facilitated the development of a Reinforcement Learning application. In this application, an agent is realized in the form of a drone designed to learn how to detect and extinguish fires in its immediate surroundings. This Reinforcement Learning agent’s ability to navigate, learn, and adapt to dynamic fire scenarios offers a cutting-edge approach to fire management and simulation.

Used Technologies

The development of the ROSHAN simulation, aimed at modeling vegetation fire spread and training reinforcement learning agents, incorporates a range of technologies, each contributing to specific aspects of its functionality. The development and testing phases of ROSHAN are conducted on Ubuntu 20.04 and 22.04.

C++ and SDL2: The primary development language for ROSHAN is C++, chosen for its performance capabilities and control over system resources, essential for simulations with high computational demands. The graphical elements of the cellular automaton, fire, and the agent are rendered using SDL2. SDL2 is selected for its efficiency in managing graphics and multimedia, which is crucial for real-time visualization of the complex dynamics within the simulation.

DearImgui: For the graphical user interface, especially for user input and controlling the simulation, DearImgui is used. This tool provides a user-friendly interface, crucial for enabling real-time adjustments and interaction with the simulation.

GDAL and OpenStreetMap Integration: The integration of the CORINE Land Cover database is achieved through the Geospatial Data Abstraction Library (GDAL), facilitating the translation of geospatial data formats. This integration allows for the inclusion of realistic landscape settings in the simulation. Furthermore, the simulation leverages OpenStreetMap, supported by a NodeJS server in the background, to enable users to select specific geographic areas for their simulations.

Python and PyTorch for Reinforcement Learning: The reinforcement learning aspect of ROSHAN is developed using Python, a language widely used for machine learning and data analysis due to its extensive libraries and community support. PyTorch, a deep learning framework, is employed for the development and training of the neural networks that power the Reinforcement Learning agent.

FireSPIN

The FireSPIN (Fire Stochastic Particle Integrator) model, as described by Mastorakos et al. [29] [23], represents a simulation for modeling the spread of wildfires, particularly in areas where wildland and urban environments intersect. This model adeptly integrates cellular automata, which simulate forest fires, with a stochastic random walk method to model the movement of firebrands and hot gases.

The fundamental concept of the FireSPIN model is the discretization of the landscape into a grid of cells. Each cell embodies a specific segment of land, characterized by unique properties like vegetation type, the presence of buildings, streets or water. The state of each cell is dynamically updated throughout the simulation, reflecting its current burning status.

A key feature of FireSPIN is the employment of two types of virtual fire particles. These particles, modeled as hot fluid particles or embers, simulate the convection and dispersion of hot gases and firebrands within each cell. Other particles are modelled to simulate the radiation heat. Their trajectory and state are meticulously designed to mimic real-world firebrand behavior, facilitating the ignition of new areas.

The FireSPIN model needs comprehensive local data concerning land and buildings, including information on vegetation types and construction materials. This allows for an accurate simulation of the diverse responses of different landscape elements to fire. One objective of the model is to replicate the propagation speed of the flame front S_f, which is approximately 0.1 times the wind speed U_w at a 10m height. The ratio S_f/U_w is influenced by variables such as the bulk density and moisture content of tree crowns.

The model's governing equations for the stochastic behavior of convection fire particles are as follows:

$$\frac{dY_{st,p}}{dt} = -\frac{Y_{st,p}}{\tau_{mem}} \tag{0.2}$$

$$dX_{i,p} = F_l U_{i,p} dt, \quad \text{where } i = 1,2 \tag{0.3}$$

$$dU_{i,p} = -\frac{(2+3C_0)}{4}\frac{u'}{L_t}(U_{i,p} - U_{w,i})dt + (C_0 \varepsilon dt)^{1/2} \mathcal{N}_i \tag{0.4}$$

Equation 0.2 delineates the decay rate of $Y_{st,p}$, a scalar value indicative of a particle's thermal intensity. A particle with $Y_{st,p} > Y_{lim}$ is capable of igniting other cells. Initially,

$Y_{st,p}$ is set to Y_{init}, a value potentially linked to the specific fuel type corresponding to the particle, though it is currently a global parameter. As depicted in Equation 0.2, the particle's thermal intensity decays according to the parameter τ_{mem}, also a global variable. Equations 0.3 and 0.4 elaborate on the particles' stochastic motion, modulated by turbulent velocity fluctuations u' and the integral length scale L_t. L_t is a global parameter, and u' is computed using the wind speed U_w and A, a parameter governing the turbulent fluctuations. The turbulent kinetic energy ε is calculated with $\varepsilon = \frac{(u')^3}{L_t}$. The particles' random positions change due to the Lagrangian description of turbulent dispersion [30], where $U_{w,i}$ represents the current wind speed in the ith direction, and $U_{i,p}$ is the current velocity component of the particle. Constants $F_l \sim 0.1$ and $C_0 \sim 2$ serve as scaling parameters, and $\mathcal{N}_i$ is a normally distributed random variable with zero mean and unit variance.

For modeling radiant heat transfer, a simplified ray-tracing method is employed, as described by the following equations:

$$dr_p = \overline{S_{f,0}}dt + \sigma_{S_{f,0}}(dt)^{1/2}\mathcal{N}_r \tag{0.5}$$

$$d\theta_p = 2\pi\mathcal{N}_\theta \tag{0.6}$$

Equation 0.5 determines the radial coordinate length change of a particle, while Equation 0.6 describes the angular change. $\mathcal{N}r$ and $\mathcal{N}\theta$ are random variables derived from a normal distribution with zero mean and unit variance. $S_{f,0}$, initially set as a fixed constant of $0.1m/s$, is now modeled with a normal distribution defined by mean $S_{f,0}$ and standard deviation $\sigma_{S_{f,0}}$, parameters specific to each cell class's vegetation type. Another parameter, L_r, influences the characteristic radiation type of the fuel. This is used to calculate τ_{mem} for radiation particles using $\tau_{mem} = \frac{L_r}{S_{f,0}}$, and $Y_{st,p}$ is then determined using Equation 0.2. The position of a radiation particle can be calculated as follows:

$$X_{1,p} = X_{1,mc} + r_p\cos(\theta),\ X_{2,p} = X_{2,mc} + r_p\sin(\theta). \tag{0.7}$$

Each cell, based on its vegetation type, exhibits distinct ignition delay times τ_{ign} and burning durations τ_{burn}. These parameters critically determine how rapidly various materials ignite. During each timestep, it is assessed for every particle whether a cell is "visited". If so, the cell begins "ticking", and if the timesteps exceed τ_{ign}, the cell commences burning for τ_{burn} timesteps. Depending on their fuel type, cells can emit varying quantities of particles, which are released evenly over the duration of τ_{mem}.

CORINE Land Cover Database

The CORINE Land Cover Database, established as part of the Copernicus Land Monitoring Service [31], has been a fundamental resource in providing harmonized land cover and land use data across Europe for over thirty years. Recognized for its continental-scale coverage and consistent data series, CORINE has played a vital role in environmental monitoring, urban planning, and climate assessments. However, its limitations, including coarse spatial resolution and certain gaps in land cover detail, led to the development of the CLC+ Backbone. This advancement significantly enhances the capabilities of the traditional CORINE database, making it more suitable for intricate applications like the ROSHAN Simulation.

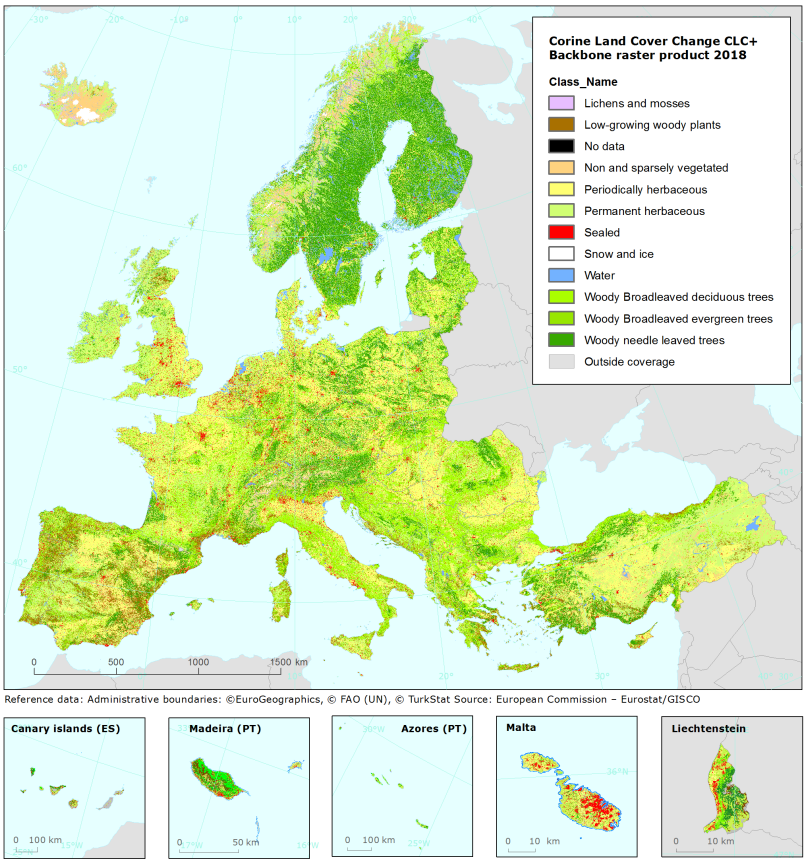

Figure 7: Coverage of the CLC+ Backbone raster product

Enhanced Spatial Resolution in CLC+ Backbone

The CLC+ Backbone [32] introduces a 10 m spatial resolution raster product, a notable improvement over the previous versions of the CORINE database. This finer spatial resolution is critical for the ROSHAN simulation, as it allows for a more detailed and granular representation of land cover. Such precision is essential in accurately modeling fire spread, particularly in areas with diverse vegetation and complex terrain.

Expanded Land Cover Classes in CLC+

CLC+ Backbone significantly expands the range of land cover classes, offering 11 basic classes in its raster product. This expansion is crucial for the ROSHAN simulation. It provides a more comprehensive categorization of vegetation types, enhancing the simulation's capability to model fire behavior across different landscapes with higher accuracy. The detailed classification includes categories such as various types of woody vegetation, herbaceous cover, and non-vegetated areas, each playing a distinct role in fire dynamics.

Integration with Sentinel Satellite Imagery

The data for CLC+ Backbone is primarily derived from multi-temporal Sentinel-2 satellite imagery. This integration ensures that the land cover information is not only detailed but also reflective of current conditions. For ROSHAN, this means the simulation can utilize up-to-date land cover data, essential for modeling and predicting fire spread in real-time scenarios accurately.

Overview of CLC+ Backbone Raster Product Classes

The CLC+ Backbone raster product offers a detailed classification of land cover types across Europe. This classification is divided into 11 basic land cover classes, each representing a unique environmental characteristic.

List of Classes

1. **Sealed**: Areas covered by impervious materials such as concrete and asphalt, typically found in urban and industrial zones.

2. **Woody - Needle Leaved Trees**: Regions predominantly covered by needle-leaved, coniferous trees.
3. **Woody - Broadleaved Deciduous Trees**: Areas primarily consisting of broadleaved trees that lose their leaves seasonally.
4. **Woody - Broadleaved Evergreen Trees**: Zones dominated by broadleaved evergreen species, retaining leaves throughout the year.
5. **Low-Growing Woody Plants (Bushes, Shrubs)**: Areas covered by shorter woody plants, including bushes and shrubs.
6. **Permanent Herbaceous**: Land covered by herbaceous vegetation that remains consistent throughout the year.
7. **Periodically Herbaceous**: Regions where herbaceous vegetation coverage varies seasonally.
8. **Lichens and Mosses**: Areas predominantly covered by lichens and mosses.
9. **Non- and Sparsely-Vegetated**: Terrain with limited or no vegetation, such as bare rock or desert landscapes.
10. **Water**: Bodies of water including lakes, rivers, and other watercourses.
11. **Snow and Ice**: Regions primarily covered by snow and ice throughout the year.
12. **Outside Area**: Areas outside the defined scope of the CLC+ Backbone raster product.

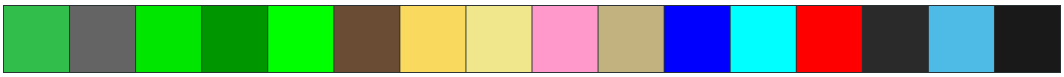

Figure 8: Cellstates as displayed in the ROSHAN simulation, from left to right: Generic Unburned(not in the model), Sealed, Woody - Needle Leaved Trees, Woody - Broadleaved Deciduous Trees, Woody - Broadleaved Evergreen Trees, Low-Growing Woody Plants (Bushes, Shrubs), Permanent Herbaceous, Periodically Herbaceous, Lichens and Mosses, Non- and Sparsely-Vegetated, Water, Snow and Ice, Generic Burning(not in the model), Generic Burned(not in the model), Generic Flooded(not in the model), Outside Area

Reinforcement Learning in ROSHAN

In the ROSHAN Simulation, the focus on Reinforcement Learning takes a central role, particularly through its implementation within an environmental context. The simulation's environment, designed to facilitate Reinforcement Learning, features an Unmanned Aerial Vehicle (UAV) as the acting agent. The UAV's primary mission within this simulation is the detection and extinguishment of fires on the map, an endeavor that represents a significant application of Reinforcement Learning in practical, real-world scenarios.

Input Parameters

The agent's input parameters are comprehensive, offering a detailed representation of the simulation environment:

Terrain: The agent receives a two-dimensional matrix representing the terrain within its visible range. During development, this range consistently encompassed a 21x21 grid with the agent centered, effectively capturing a detailed snapshot of the surrounding terrain types.

Fire Status: Parallel to terrain awareness, the agent is also fed a two-dimensional matrix of the same visual scope. This matrix, however, encodes information about the presence or absence of fire in each cell, giving the agent a clear indication of immediate fire threats.

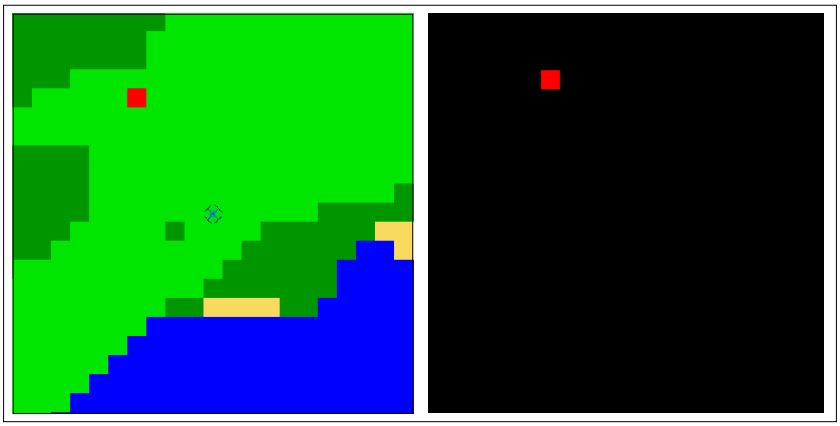

Figure 9: **Left:** Terrain input, the agent in the middle of the image is not given as information. **Right:** A binary map of the same size that shows burning cells.

Velocity: The agent is provided with its current velocities in both the X and Y directions, with these values normalized between -1 and 1.

Position: The agent has knowledge of its relative position within the overall map. This is defined with the top-left cell as (0,0) and the bottom-right as (1,1), allowing for continuous positional tracking as the agent moves across the grid map.

Map Overview (Discontinued): Initially, the agent was designed to receive the entire map as a two-dimensional matrix input. This feature would mark revealed fires on the map, akin to the ”Fire Status” matrix. The idea behind this input was that the agent could remember spotted fires on the map. However, due to poor performance, this input was later discontinued.

Normalization was applied to all input values unless specified otherwise, ensuring data uniformity and facilitating the agent's processing of this information. Each of the above inputs was collected over the last couple of timesteps.

Network Output

The network output comprises two components. First, the velocity controls, which encompass velocity change in the X and Y directions, are used to navigate the agent through the environment. Second, the water release command, an output value that dictates the precise moments when the agent releases water. The water release command will flood the cell the agent is currently on and extinguish the cell. It will also extinguish every particle that is present on that cell. A cell stays flooded for several simulation steps and than go back to normal.

Network Architecture

The network architecture is composed of three primary modules: Inputspace, Actor, and Critic. Each module is tailored to handle specific aspects of the data processing and decision-making process in a reinforcement learning framework. It is important to note that there are two distinct Inputspace Networks for the Actor and the Critic.

Inputspace Module: This module serves as the initial processing unit, designed to handle various input types like terrain, fire status, velocity, maps, and position. It employs 2D convolutional layers, followed by batch normalization, to extract and standardize spatial features from the inputs. The convolutional layers are parameterized by kernel size (3x3), stride (1x1), and padding (1x1), facilitating detailed feature extraction. The

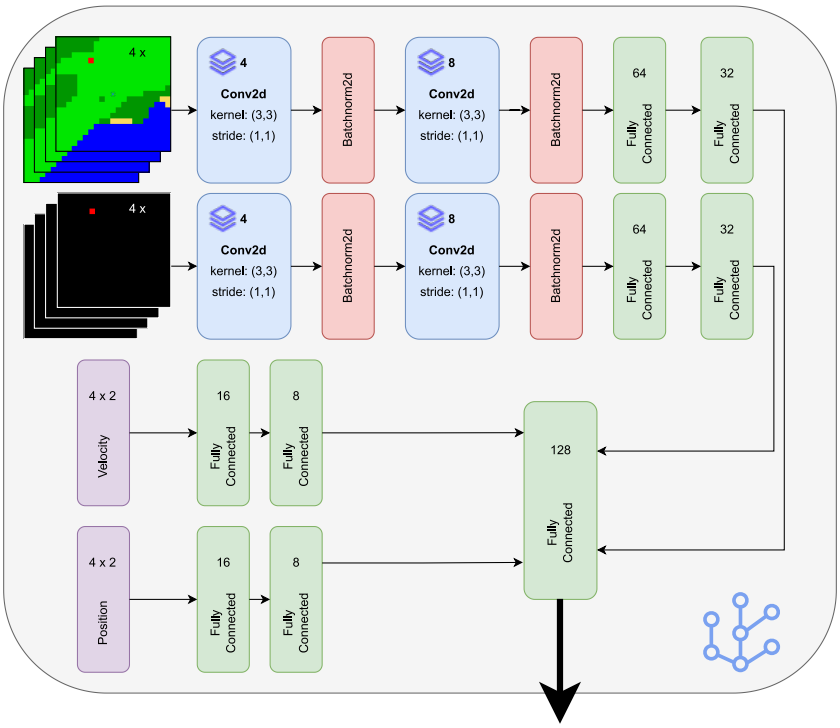

Figure 10: Inputspace Module.

data is then flattened and passed through linear dense layers to reduce dimensionality, preparing it for integration in the Actor and Critic modules.

Actor Module: This module is responsible for generating action outputs. It integrates the processed input from the Inputspace module and predicts actions using linear layers with nonlinear activations (tanh and sigmoid). The module outputs both the action values and their variances, with the variance computed using a parameterized log standard deviation.

Critic Module: Similar to the Actor module, the Critic module leverages the processed input from the Inputspace module. Its primary function is to estimate the value of input states, utilizing a linear layer for this purpose. The value estimations provided by the Critic module are integral to the reinforcement learning process, guiding the optimization of the policy network.

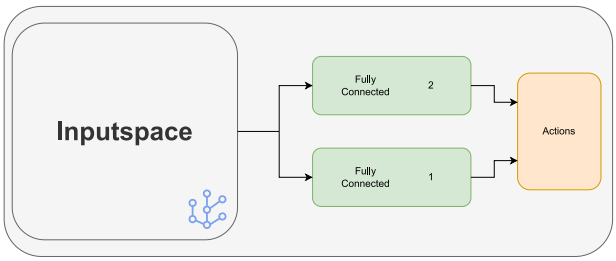

Figure 11: Actor Module

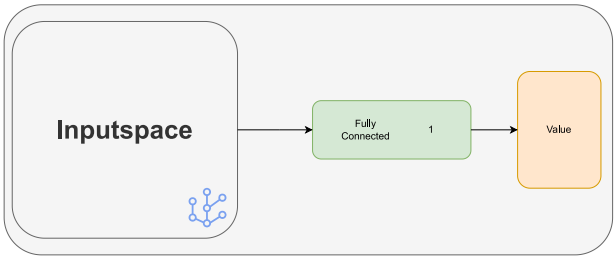

Figure 12: Critic Module

Reward Calculation

The reward mechanism in ROSHAN is multifaceted, focusing on various aspects of the agent's performance. Before training, the collected rewards are normalized subtracting their mean and dividing with their standard deviation.

Boundary Awareness: The agent is penalized for leaving the designated area, with an increased penalty if the drone reaches a terminal state outside the grid. This encourages the drone to operate within the operational boundaries.

$$r_{bnd} = \begin{cases} -2 \times \text{distance\_from\_grid} - 10 & \text{if outside\_boundaries and drone\_terminal} \\ -2 \times \text{distance\_from\_grid} & \text{if outside boundaries} \end{cases} \tag{0.8}$$

Fire Proximity: The agent is rewarded for advancing towards the nearest fire within its vicinity. Conversely, the agent incurs penalties when it moves away from a nearby fire. These rewards are scaled based on the nearest fire's proximity $\Delta fire\_dist$, promoting efficient fire management. This mechanism ensures the agent is motivated to engage promptly and effectively with fires in its proximity.

$$r_{prox} = 0.05 \times \Delta fire\_dist \tag{0.9}$$

Fire Extinguishing: The agent receives positive rewards for discovering and extinguishing fires.

$$r_{detect} = 2 \quad \text{if last\_near\_fires} = 0 \text{ and near\_fires} > 0 \tag{0.10}$$

$$r_{extinguish} = \begin{cases} 10 & \text{if fire\_extinguished and drone\_terminal} \\ 5 & \text{if fire\_extinguished and near\_fires} = 0 \\ 1 & \text{if fire\_extinguished} \end{cases} \tag{0.11}$$

The total reward is determined by aggregating these individual sub-rewards.

$$r_{total} = r_{bnd} + r_{prox} + r_{detect} + r_{extinguish} \tag{0.12}$$

Usage

Overview

The ROSHAN simulation, designed for intuitive and user-friendly interaction, operates in two distinct modes. Users can engage with it as a standalone application developed in C++ or integrate it as a module within Python environments.

Launching and Model Selection

Upon launching ROSHAN, users are greeted with a window that presents model options. Currently, the options include test implementations of the Game of Life and the advanced FireSPIN model. Selecting ”FireSPIN” initiates a new window, prompting users to choose or modify the initial map. The choice here lies between generating a map with a uniform distribution of generic cells or loading a pre-saved map.

Map Configuration and Control

Once a map is selected, it displays alongside essential control elements. Users can adjust simulation parameters, including settings for Convective and Radiation Particles and wind conditions. If a uniform generic cell map is chosen, users have the option to modify specific cell properties. Conversely, if the map originates from the CORINE database, these properties are predetermined by the cell class.

Simulation Dynamics

The ”Simulation Controls” window offers functionalities to start, pause, and modify the simulation. Users can set the simulation time, introduce delays for observation purposes, and view the current Frames Per Second (see Figure 13). This real-time control and feedback allow users to monitor and adapt the simulation dynamically.

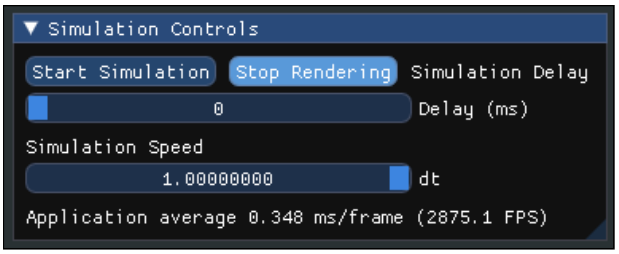

Figure 13: Simulation Control Window

Integration with External Data

A notable feature of ROSHAN is its integration with external data sources. Through the ”File” menu, users can open a browser window displaying an OpenStreetMap instance. Here, users can select a specific area for import from the CORINE database into the simulation grid. However, it’s important to note that only one area can be marked at a time.

Saving and Loading Maps

Additional options under the ”File” menu (see Figure 14) include saving the current CORINE map to disk, loading previously saved maps, resetting the GridMap to its initial state, and toggling between uniform and CORINE maps.

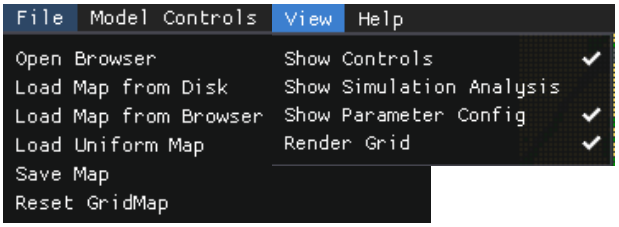

Figure 14: The simulation features a menu with ’File’ and ’View’ options, offering functionalities for loading and saving the current map, as well as options for visualizing information and toggling various control windows.

Model and View Customization

In the ”Model Controls” menu, users can enable or disable Convective and Radiation Particles within the FireSPIN simulation. The ”View” menu offers options to toggle windows for ”Simulation Parameters” and ”Simulation Controls”, access information about the simulation’s current state, and adjust the map’s visual representation by toggling a grid pattern overlay (see Figure 14).

Reinforcement Learning Integration

When ROSHAN is operated in Python with the Reinforcement Learning environment, an additional ”Reinforcement Learning Controls” tab becomes available. This feature includes controls for starting and stopping agent training, resetting agents, and accessing the ”Drone Analysis” button, which provides insights into the agent’s current status and accumulated rewards. There is also a visible agent appreaing at a random location on the map.

Interactive Map Features

The interactive map within ROSHAN enhances user engagement. Users can move the map using the right mouse button, zoom in and out with the mouse wheel, and interact with cells to ignite or extinguish them with a click on the left mouse button. Pressing the mouse wheel will open a popup window of the cell, informing the users about specific cell properties. In Reinforcement Learning mode, the agent can be controlled using the ”WASD” keys, and the spacebar is used to deploy the extinguishing payload.

Figure 15: Popup shows the cell information. This includes current position in the grid aswell as the current cell state, it’s dimensions and internal model parameters.

Results and Discussion

Simulation Output

In this section, we will discuss the output of the ROSHAN simulation model under varying conditions and at different time steps. It’s important to note that no evaluation of the fire simulation itself was undertaken within this work. Typically, such evaluations

are conducted either by comparing the results with existing simulations like FARSITE or by matching them with real-world fire incidents. It is also crucial to emphasize the importance of comparing with real fires or advanced simulations for future system development. To compensate for the lack of proper evaluation, this paper focuses on presenting the model output under various scenarios and compares it with the output from the previous FireSim paper, because the newer implementation of FireSim does not offer general fire simulations but evaluates the simulation with a real fire occuring in the USA, making a direct comparison impractical. So, the comparison is made with the model from the previous paper, which only differs slightly from the implemented model, as can be seen in figures 16, 17 and 17. This approach is not comprehensive due to the diversity of adjustable parameters, but it should still demonstrate the basic dynamics of the model and make it understandable for the reader how fire spreads under different conditions in the model.

This is a list of the most important parameters to be adjusted in the ROSHAN simulation, noting that cell parameters must be set differently for each terrain type. In this work, the parameters for radiation particles were primarily adopted from the latest FireSPIN paper. Some parameters are derived from the cell class, such as the number of particles a burning cell emits and the range of radiation. These parameters were not predefined and hence were initially estimated. Through observation and iterative adjustments during the simulation, a refined understanding was achieved. For instance, trees, represented by certain cell types, burn longer and emit more particles compared to grasslands. However, their particles travel further, resulting in tree-like cells being more likely to ignite distant cells, while grassland-like cells tend to ignite cells in their vicinity.

- **Convective Particle Parameters:**
  - Y_{st}: Hotness of the Particle
  - Y_{lim}: Duration the particle can cause ignition
  - Fl: Scaling Factor
  - C_0: Constant, typically near 2
  - τ_{mem}: Particle Lifetime in seconds
  - L_t: Height of emitting source in meters
- **Radiation Particle Parameters:**
  - Y_{st}: Hotness of the Particle

  - Y_{lim}: Duration the particle can cause ignition

- **Cell Parameters:**
  - τ_{ign}: How fast does the cell burn in seconds, additionally, this parameter fluctuates by 20% to simulate natural fluctuations in the terrain.
  - *Burning Duration*: How long does the cell burn in seconds
  - *Cell Size*: Dimensions of each cell in m^2
  - *Particles*: Number of particles each cell has, and how the cell emits them (all at once or correlated to the burning time)
- **Wind Parameters:**
  - U_w: The 10m wind speed in m/s
  - *Angle*: Angle of the wind
  - A: Component of the wind speed in the 1st direction

Radiation Particles

The propagation of radiation in fire spread models plays a critical role in understanding and predicting fire dynamics. In the context of the FireSPIN simulation, radiation propagation exhibits unique characteristics that significantly impact the ignition of cells and the overall fire spread pattern. The original implementation of FireSPIN featured a radiation propagation mechanism that was somewhat chaotic in its spread but had a pronounced effect on igniting cells, as depicted in Figures 16, 18, and 18.

In the revised implementation of FireSPIN, fire spread through radiation in a more uniform manner, especially when considering a uniform distribution of cells, compared to the original model (see Figure 18). The authors of the FireSPIN paper mentioned that radiation particles were incorporated for completeness, but they were not utilized during their simulation. The rationale behind this was the negligible impact of radiation particles on the simulation outcomes, as illustrated in Figure 20.

Under the specified parameters and with a uniform distribution, it was not possible to achieve significant fire spread with just one burning cell. The chain reaction initiating a fire, propagated solely through radiation, commenced only when multiple nearby cells were ignited. Naturally, this behavior is dependent on the set parameters.

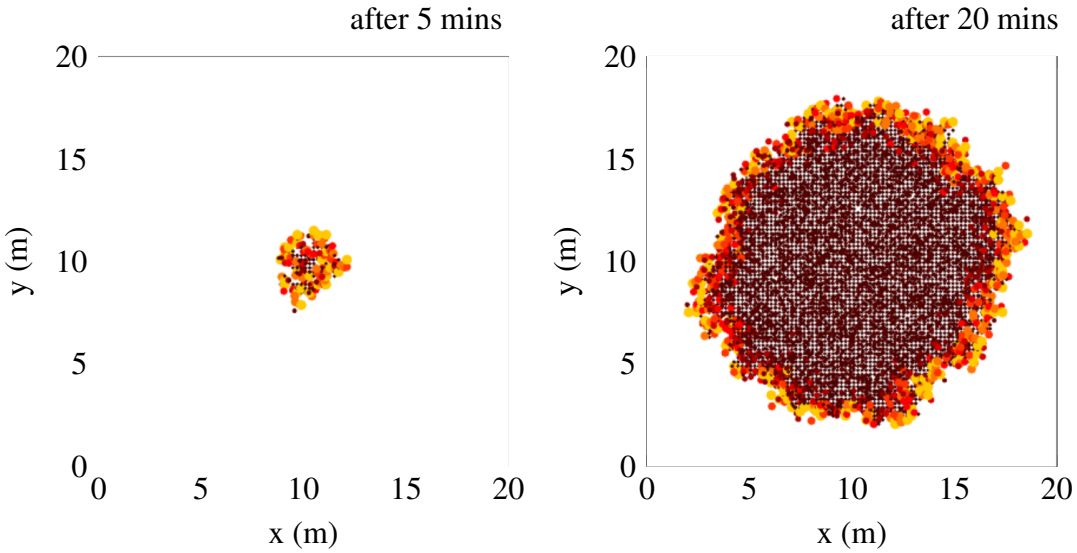

Figure 16: Radiation propagation of the original implementation. There is no simulated wind speed and ignition point was in the center of the grid. The grid consists of 100x100 cells with each cell measuring $2mx2m$ in a uniform vegetation. τ_{ign} was set to $10s$ and Y_{lim} to 0.2.

Convective Particles

The propagation of fire through convective particles is a crucial aspect of fire spread in this simulation, closely resembling the behavior described in the original FireSPIN implementation. The key characteristic of this model lies in its treatment of particle emission, which, despite varying in quantity from the original implementation, does not significantly alter the overall fire spread dynamics. Therefore, a comparative analysis with the original paper is not pursued here.

Convective particles in this model predominantly spread via wind, with wind direction playing a pivotal role in their movement. Counter-directional spread against the wind is only feasible under conditions of very mild wind, making wind direction a critical factor in predicting the path and speed of fire spread. In uniform terrains, a typical cone-shaped propagation pattern of the fire front is observed, indicative of the wind's influence on the fire's direction and intensity.

A salient feature of the convective particles is their dependency on underlying parameters that govern their behavior. One such parameter, τ_{mem}, significantly impacts the fire front's propagation. This parameter determines the lifespan of a particle before it

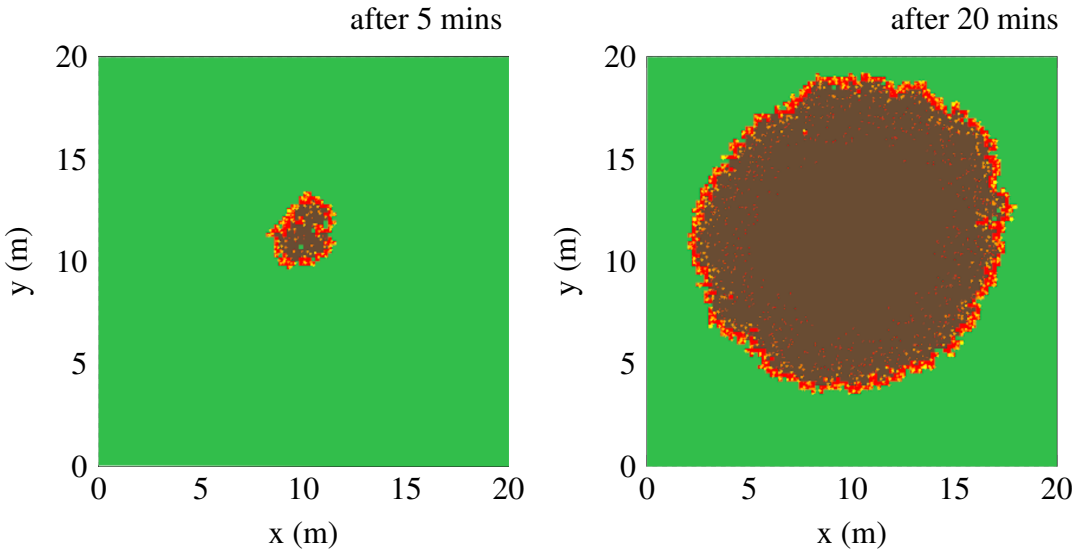

Figure 17: Radiation propagation of the original implementation used in ROSHAN. The parameters were set according to those in figure 16.

extinguishes. A visual representation of this effect can be observed in Figure 19, where the influence of τ_{mem} on fire spread is illustrated.

When τ_{mem} is set to a sufficiently high value, it enables fire to spread over water bodies or across other non-combustible materials. This intriguing phenomenon is depicted in Figure 21. In this scenario, the particles possess the capability to traverse over water and ignite cells on the opposite side before they extinguish. In contrast, lower values of τ_{mem} significantly limit this capability, leading to a scenario where non-combustible passages effectively halt the fire's progression.

In addition to the impact of the τ_{mem} parameter, the fire spread dynamics in the model are also significantly influenced by other factors, notably wind speed. When wind speed is high, it creates conditions akin to those observed with a higher τ_{mem} setting. The enhanced wind velocity can carry convective particles across natural barriers like water bodies, aiding in the fire's propagation to previously unreachable areas. Furthermore, the parameters F_l and τ_{ign} play substantial roles in the model. The τ_{ign} parameter, representing the ignition delay, determines the time taken for a cell to ignite after being reached by a particle. A shorter τ_{ign} leads to quicker fire spread as cells ignite more rapidly, whereas a longer τ_{ign} can slow down the overall pace of the fire spread, potentially stop the fire propagation as cells take too long to incinerate.

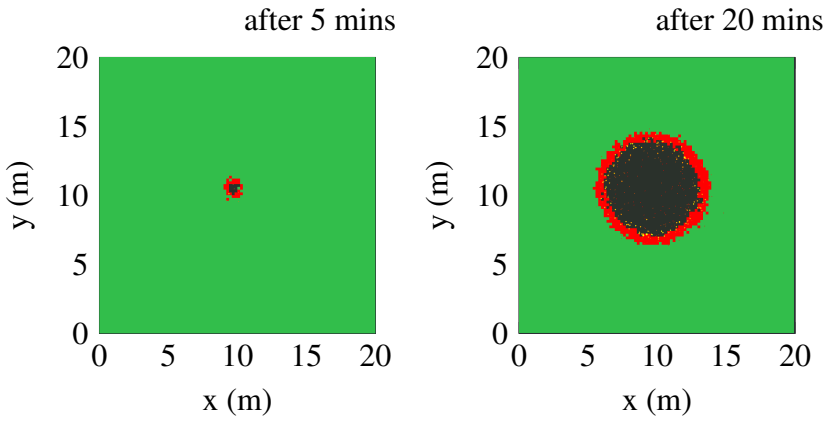

Figure 18: Radiation propagation of the updated model currently used in ROSHAN. The parameters were set according to those in figure 16. Additionally this implementation emmited 4 particles over a timespan of 120s. The chain reaction initiating a fire, propagated solely through radiation, commenced only when multiple nearby cells were ignited.

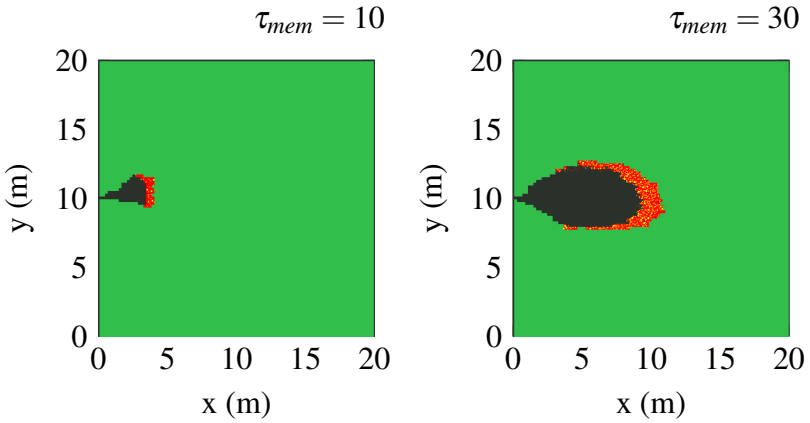

Figure 19: Both pictures are different simulations with only the τ_{mem} parameter changing. They were both ignated at the same cell and simulated over 5 minutes. The longevity of a particle, as dictated by τ_{mem}, directly affects how far and fast the fire can spread. Particles with a longer existence, as per a higher τ_{mem} value, have the potential to ignite new areas further from the original fire source, leading to a more widespread and potentially erratic fire spread. Conversely, a lower τ_{mem} results in particles extinguishing more rapidly, potentially leading to a slower and more contained fire progression.

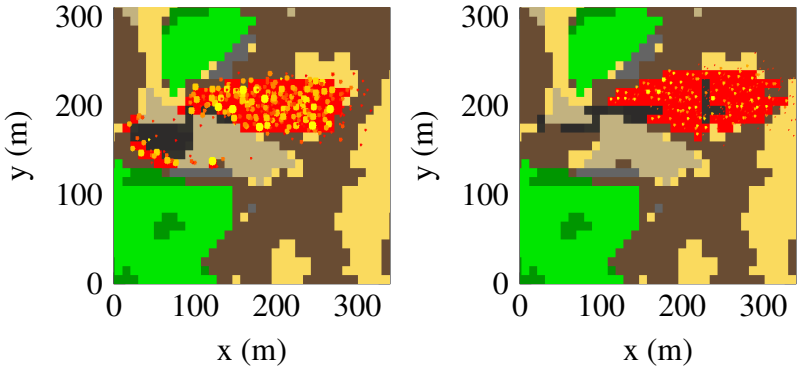

Figure 20: Comparison of fire spread simulations over a 20-minute period. The left side depicts the model when a cell is ignited with both Convective and Radiation Particles, while the right side illustrates the simulation using only Convective Particles. In both simulations, the same cell was ignited.

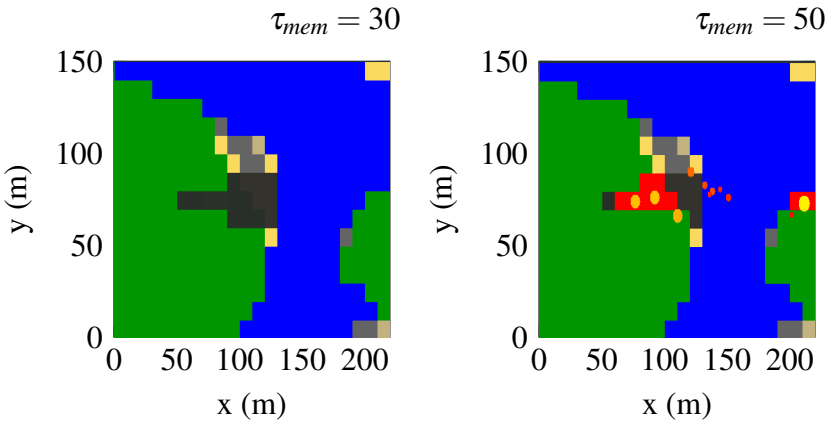

Figure 21: When τ_{mem} is set to a value of 30, the fire front is incapable of bridging the water, as depicted on the left side of the illustration. Conversely, setting τ_{mem} to 50 enables the fire to propagate effortlessly to the other side of the river.

Agent Performance

The performance of the agent was assessed through a series of controlled experiments designed to validate the feasibility of the approach. The agent underwent a limited scope of testing, which primarily focused on demonstrating the general effectiveness of the method.

A total of 10 tests were conducted, with training and evaluation occurring on distinct maps(see Figure 22). The training environment comprised a relatively small grid of 420 cells (20 x 21 cells), representing an area of 4.2 hectares. The initiation of each training episode involved randomly positioning five fires across the map and placing the drone at a random location. Notably, one of the fires was always positioned directly beneath the drone, ensuring that the agent had a favorable opportunity to receive a positive reward in each episode. Training sessions lasted for 15 minutes each.

The PPO parameters were set as follows:

- Batch Size: 2048
- Mini Batches: 64
- Learning rate: 0.00003
- Discount factor (γ): 0.99
- GAE parameter (λ): 0.9
- Number of epochs (K_epochs): 4
- Clipping parameter (ε): 0.2

Each of the 10 networks was evaluated over 51 episodes, with three possible outcomes for each episode:

1. All fires extinguished (goal reached).
2. Over 30% of the map burnt (agent failed).
3. The agent remained outside the map boundaries for too long (agent failed).

It is pertinent to note that a complete burnout of the map was never observed, making ”agent failed” synonymous with the agent straying off the map. The collected data included the number of successful and unsuccessful episodes, the average reward, and the average number of steps taken by the agent until the episode concluded.

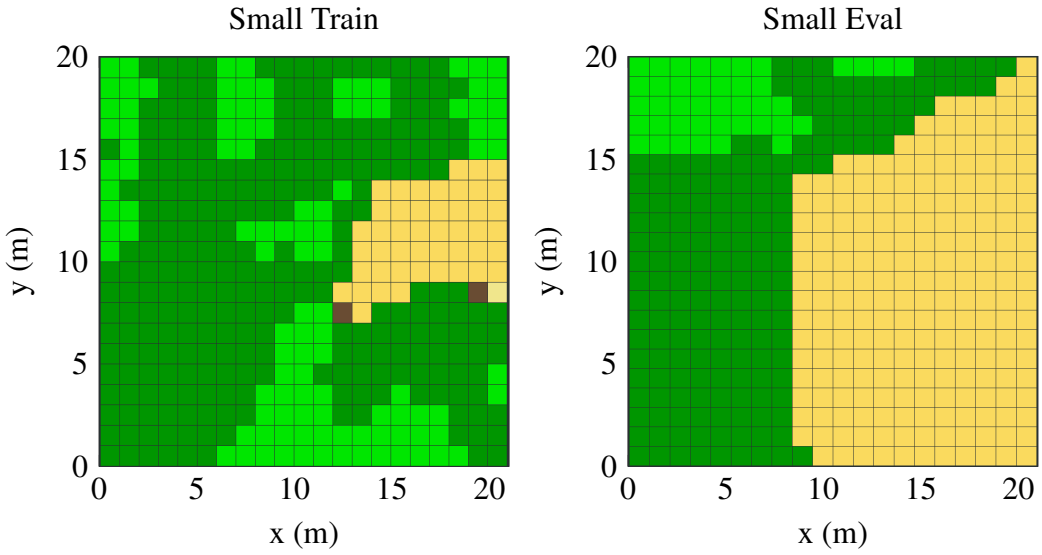

Figure 22: The maps used for training and evaluation. Both have the same size of 4.2 hectares.

Networks 1, 3, 4, 5, and 7, despite achieving some level of success in reaching their goals, also exhibited instances of failure. A notable pattern in these networks was the occurrence of ”suicide” by the agents, characterized by rapidly moving out of the simulation map. This behavior indicates potential issues in the decision-making logic of the agents in these networks, leading them to make detrimental choices.

Networks 2, 6, 8, 9, and 10, in contrast, showed superior performance, with no recorded failures and a high rate of successfully reaching their goals. However, it’s observed that some of these well-performing networks, particularly Networks 2, 8, and 9, took longer times to reach their goals. This delay was attributed to the agents getting stuck at corners of the map or, in one notable instance, between two burning cells. Interestingly, Network 6 stands out not only for its lack of failures but also for being the fastest overall. This suggests that the agents in Network 6 were more adept at navigating the simulation environment without getting trapped or resorting to ”suicide.”

In conclusion, it can be stated that the application of Reinforcement Learning to simple agents and environments in this study was successfully demonstrated. The results show that Reinforcement Learning agents can be effectively trained to extinguish fires in simulated environments. Particularly noteworthy is the performance of networks 2, 6, 8, 9, and 10, which achieved their goal without failure, although some networks required longer times to achieve their objectives. The findings can serve as a basis for further

Table 1: Agent Training Results

Network No.	Average Reward	Average Time	Failed	Reached Goal
1	10.12	141.13	31	20
2	30.02	607.7	0	51
3	16.01	126.5	22	29
4	10.54	101.07	30	21
5	10.83	102.15	28	23
6	30.86	82.11	0	51
7	15.34	90.86	21	30
8	30.98	113.27	0	51
9	30.3	354.62	0	51
10	31.07	235.50	0	51

research aimed at developing more powerful and adaptable agents for more complex and realistic environments.

Challenges

The development of ROSHAN, particularly the integration and utilization of the SDL, presented several performance-related challenges. These issues primarily stemmed from lack of prior experience with SDL. The implementation process, especially regarding the model's particle system, initially led to significant performance problems. Fortunately, most of these issues were successfully addressed and resolved in subsequent development stages. Another major challenge involved the incorporation of terrain data from the CORINE database. While the database provided various classes representing different types of terrain, the parameters derived from these terrain types were not initially defined. This required extensive testing to adjust and optimize these parameters to acceptable ranges, ensuring they accurately reflected the characteristics and behaviors of the respective terrains in the simulation. From the outset, it was clear that the scope of this project would only allow for the realization of a rudimentary Reinforcement Learning agent. However, designing even this basic agent proved to be a complex task. Various errors within the codebase adversely affected the agent's learning capabilities. Identifying and rectifying these errors was a time-consuming process, taking up more time than initially planned.

Review

Current Limitations

While significant progress has been made in addressing some performance issues in ROSHAN, there remains considerable scope for optimization. Particularly, when dealing with larger maps and in the context of Reinforcement Learning operations, the execution time is comparatively slow. Another performance bottleneck arises from the intrinsic nature of the simulation model. As the size of the map increases, the number of particles emitted over time also grows exponentially. Since each particle requires computation, the simulation slows down markedly with a large number of particles. This challenge was also highlighted in the foundational paper of the FireSPIN simulation. Moreover, the current model has limitations in its representation of crucial parameters for fire propagation. Notably, the slope of the terrain, a significant factor in wildfire spread, is not currently considered. Additionally, the model does not account for the moisture level of the region, which can greatly influence fire behavior. While wind and terrain, two primary factors influencing wildfire spread, are incorporated into the current model, there is room for improvement in their dynamic representation. The way wind is modeled in ROSHAN could be made more dynamic, reflecting the variable and often unpredictable nature of real-world wind patterns. Similarly, the parameters defining terrain characteristics could be optimized further, enhancing the model's ability to accurately simulate fire behavior in diverse topographical settings.

Future Enhancements

One of the key enhancements in future iterations of ROSHAN involves the refinement of wind simulation within the model. Wind plays a crucial role in the behavior of wildfires, influencing their direction and speed of spread. Future enhancements could focus on incorporating more complex and dynamic wind models that can better mimic real-world wind patterns and their impact on fire behavior. The terrain is a critical factor in the spread of wildfires, with elevation and slope significantly affecting fire behavior. The current model in ROSHAN requires further refinement in terms of terrain parameters. Future enhancements may incorporate a more detailed and accurate representation of terrain, including slope integration, which is vital for simulating fire spread in hilly or mountainous areas. This would involve processing of terrain data and integration into the fire spread model effectively. The efficiency of the ROSHAN simulation, particularly in terms of computation time, is another area marked for possible improvement.

For validation and enhancement of the model's credibility, a qualitative comparison with other fire simulations or real wildfire incidents is essential. This involves analyzing how well ROSHAN's predictions align with real-world fire behaviors and/or outcomes from other established models. These comparisons will help identify discrepancies and areas of improvement, leading to a more robust and reliable simulation tool. The current version of ROSHAN supports a simple UAV with a basic reward function for extinguishing smaller fires. Future enhancements could see the Reinforcement Learning agent evolve to systematically explore areas and locate fires more effectively. In addition to UAVs, the incorporation of Unmanned Ground Vehicles (UGV) in the ROSHAN simulation could be a possibility. UGVs, with their realistic firefighting capacities, could provide a more practical approach to fire suppression in the model. Their integration would allow the simulation to encompass a broader range of firefighting strategies and tactics, enhancing the overall utility of the ROSHAN simulation in fire management and emergency response planning. Regarding the Reinforcement Learning aspect of this paper the challenge of training agents in larger and more complex environments remains. Future work should focus on improving the mapping and exploration capabilities of the agents. One possible approach could be to provide the agents with a larger map, which would require an adaptation of the agent's learning abilities. In earlier versions of this work, such an approach was already considered by using the entire map as input, but it was not pursued further due to limitations in the agent's learning abilities at the time. Furthermore, the current reward function should be revised to provide a stronger incentive for agents to explore their environment. This is particularly important as tests on larger maps showed that agents tended to get stuck in certain places. An improved reward function could help address this problem by encouraging agents to more actively explore their environment and respond more effectively to changes in the scenario.

Utilization of ChatGPT

In the evolving landscape of academic research and writing, the integration of artificial intelligence tools has become increasingly prevalent. This paper embraces this technological advancement by incorporating ChatGPT, an AI language model developed by OpenAI [33], as a pivotal tool in its composition. The objective of this chapter is to transparently outline the methodology employed in utilizing ChatGPT and to clearly identify the sections within the paper that were predominantly crafted with its assistance.

ChatGPT served as an instrumental resource, offering assistance in various stages of the paper's development. From generating initial ideas and structuring content to refining language and ensuring coherence, its role was multifaceted. However, it is important to emphasize that the core ideas, hypotheses, and critical analyses originated from human intellect and academic inquiry, while ChatGPT primarily aided in articulation and presentation.

The following sections will detail the specific contributions of ChatGPT to this paper. We will explore how this tool was employed to enhance the writing process, ensuring that the final product not only adheres to academic standards but also reflects a synergy between human intellectual effort and AI efficiency.

General Method

For the development of this paper, I leveraged the capabilities of ChatGPT 4 Pro Version, a state-of-the-art artificial intelligence model by OpenAI []. This advanced tool was not just utilized in its standard form; instead, it was customized to cater specifically to the needs of this research paper. The customization of ChatGPT involved several critical components, each playing a vital role in shaping the tool's contribution to our paper. These components include the GPT's name, description, instructions, and its knowledge database.

Customized GPT

GPT Name The customized GPT was assigned a unique name. This name served as an identifier, distinguishing our tailored version from the general ChatGPT model.

Description The description of the customized GPT encompassed its primary functions, capabilities, and the scope of its application within our research.

Instructions The instructions for the customized GPT were pivotal in guiding its functionality. These instructions act as a directive, informing the AI about the specific tasks it needs to perform, the style of writing to be adhered to, and the manner in which it should process and present information.

> ”ROSHAN Writer is a conversational and friendly AI assistant, designed to help with academic writing, focusing on structuring and refining term papers. It provides concise and direct answers, utilizing its own training and a knowledge base provided by the user. Equipped to handle LaTeX formatting and occasional German words, it will ask for clarification when necessary for accuracy. ROSHAN Writer uses jargon typical in computer science, aligning with the user’s field. Responses are professional yet approachable, aiding in creating well-structured, coherent academic documents. The knowledge base now includes the files ’Masterseminar_Feuer.pdf’, ’Structure.tex’, and ’Concept.docx’, expanding available resources for reference.”

Knowledge Database Perhaps the most significant component was the knowledge database integrated into the customized GPT. This database included a collection of documents relevant data that were essential to the topic of our research. By having direct access to this curated information, the GPT was able to draw upon a wealth of specific and relevant knowledge, ensuring that its contributions were accurate, well-informed, and contextually appropriate for the paper.

As outlined in the customized instructions, the knowledge database of ROSHAN Writer was comprised of three distinct files. Initially, I formulated the overarching structure of this project and uploaded it as the ’Structure.tex’ file. Additionally, I provided a summary of my work in the ’Concept.tex’ file, thereby equipping the GPT with a foundational understanding of the subject matter. The final piece of this trio was a paper I had previously authored on this topic, a German document titled ’Masterseminar_Feuer.pdf’.

Interaction with ROSHAN Writer

The process typically began with a clear directive to the GPT about the specific chapter or section in focus. For instance, if the task was to create a subchapter on radiation

propagation, I would instruct the GPT as follows: ”I need you to draft a subchapter about radiation propagation. Please write in English, utilizing both your knowledge database and the information I provide. Aim for a fluent, continuous text and format it in LaTeX.” This instruction was crucial in setting the tone and requirements for the AI-generated content.

Following this, I would provide my own description of the chapter, usually in German, since it is my mother tongue. The following should serve as an example for this method used for the section 'Radiation Particles'.

> ”Die Strahlung propagiert kreisförmig um die Zelle herum. FireSPIN hatte in ihrer ursprünglichen Implementierung eine Strahlungspropagierung die etwas chaotischer in Ihrer Ausbreitung war, dafür aber auch einen Größeren Effekt auf das Entzünden der Zellen hatte(siehe Abbildung 16, 17 & 18). In der Neuimplementierung von FireSPIN breitet sich Feuer allein durch die Strahlung bei uniformer Verteilung der Zellen gleichförmiger aus als die Originalimplementierung(siehe Abbildung 18). Im FireSPIN Paper beschreiben die Autoren dass sie die Strahlungspartikel nur ”der vollständigkeit halber” implementiert haben, diese während ihrer Simulation allerdings nicht nutzen. Der Grund dafür ist, dass sich die Strahlungspartikel nicht wesentlich auf das Simulationsergebnis auswirken(siehe Abbildung 19 & 20). Bei uniformer Verteilung war es unter den angegebenen Parametern auch nicht möglich mit nur einer brennenden Zelle dieses Ergebnis zu erzielen, erst wenn mehrere in der Umgebung befindliche Zellen gebrannt haben startet die Kettenreaktion und ein Feuer, das rein durch Strahlung propagiert breitet sich aus. Selbstverständlich hängt auch das von den eingestellten Parametern ab, in dieser Arbeit wurden maßgeblich die Parameter aus dem letzten FireSPIN Paper für die Strahlungspartikel übernommen. Andere Parameter werden durch die Zellklasse abgeleitet, z.b. wie viele Partikel eine brennende Zelle ausstößt und wie hoch die Strahlungslänge ist. Diese Parameter waren nicht vorgegeben und wurden deshalb zunächst geschätzt und durch Beobachten der Simulation nach und nach eingestellt. So brennen Bäume länger und stoßen mehr Partikel aus als Wiesen, aber ihre Partikel werden auch weiter getragen, was sich in der Simulation dann so auswirkt, dass baumartige Zellen eher in der Lage sind weiter entfernte Zellen zu entflammen und wiesenartige Zellen eher Zellen in ihrer Nähe entflammen.”

Iterative Refinement

After the initial text generation by the GPT, the refinement phase commenced. This was

a critical stage where I closely reviewed the AI-generated text. One common tendency of the GPT that needed addressing was its repetitive nature. Sections where the content was reiterated unnecessarily were pruned for conciseness and clarity.

Another aspect that often required modification was the GPT's penchant for using complex language and bullet points. In these instances, I would intervene to simplify the language and transform bullet points into a more narrative style, ensuring the text was accessible and flowed naturally.

Finalization and Contextual Enhancement
Once the AI-generated text underwent initial corrections, the next step was to infuse the chapter with additional context. This involved integrating more detailed information, perspectives, or data relevant to the topic. The same iterative process of review and refinement was applied, where I would manually adjust and enhance the content to ensure it met the desired quality and depth.

Through this meticulous process of instruction, iteration, and personal intervention, the customized ChatGPT agent effectively contributed to the paper's development.

Development of Sections Through Initial ChatGPT Generation

In the composition of this paper, certain segments were not initially outlined by me but were instead first generated by ChatGPT and subsequently curated and augmented. This approach was applied to the sections titled 'Wildfire Models', 'Cellular Automata', and 'Brief Introduction to Reinforcement Learning'.

Initial Generation by ChatGPT For these specific sections, the initial drafts were entirely produced by the ROSHAN Writer, the customized version of ChatGPT. This process began without predefined inputs from my side, allowing the AI to leverage its extensive knowledge database and programming to construct foundational content for these topics. The decision to let ChatGPT take the lead in these segments was driven by the desire to explore the depth and breadth of insights the AI could offer based on its integrated resources.

Post-Generation Curation and Enhancement Once ChatGPT provided the initial drafts, the subsequent phase involved my active engagement in curating, correcting, and expanding the content. This process was akin to the previously described method of iterative refinement.

Bibliography

[1] N. Bénichou, M. Adelzadeh, J. Singh, I. Gomaa, N. Elsagan, M. Kinateder, C. Ma, A. Gaur, A. Bwalya, and M. Sultan, "National guide for wildland-urban-interface fires: guidance on hazard and exposure assessment, property protection, community resilience and emergency planning to minimize the impact of wildland-urban interface fires," 2021, 196 p. [Online]. Available: https://doi.org/10.4224/40002647

[2] S. L. Manzello, K. Almand, E. Guillaume, S. Vallerent, S. Hameury, and T. Hakkarainen, "The Growing Global Wildland Urban Interface (WUI) Fire Dilemma: Priority Needs for Research," *Fire Saf J*, vol. 100, 2018.

[3] Q. Huang, A. Razi, F. Afghah, and P. Fule, "Wildfire Spread Modeling with Aerial Image Processing," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2020, pp. 335–340.

[4] S. Li and T. Banerjee, "Spatial and temporal pattern of wildfires in california from 2000 to 2019," *Scientific Reports*, vol. 11, no. 1, p. 8779, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-88131-9

[5] Y. Ban, P. Zhang, A. Nascetti, A. R. Bevington, and M. A. Wulder, "Near Real-Time Wildfire Progression Monitoring with Sentinel-1 SAR Time Series and Deep Learning," *Scientific Reports*, vol. 10, no. 1, p. 1322, 2020. [Online]. Available: https://doi.org/10.1038/s41598-019-56967-x

[6] M. P. Thompson, D. E. Calkin, J. W. Gilbertson-Day, and A. A. Ager, "Advancing effects analysis for integrated, large-scale wildfire risk assessment," *Environmental Monitoring and Assessment*, vol. 179, no. 1–4, p. 217–239, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10661-010-1731-x

[7] M. Pereira, L. Fernandes, S. Carvalho, R. Bessa Santos, L. Caramelo, and A. alencoão, "Modelling the impacts of wildfires on runoff at the river basin ecological scale in a changing Mediterranean environment," *Environmental Earth Sciences*, vol. 75, 02 2016.

[8] J. Sun, W. Qi, Y. Huang, C. Xu, and W. Yang, "Facing the Wildfire Spread Risk Challenge: Where Are We Now and Where Are We Going?" *Fire*, vol. 6, no. 6, 2023. [Online]. Available: https://www.mdpi.com/2571-6255/6/6/228

[9] J. Meine, "Modellierung von Bränden mit zellulären Automaten," 2023.

[10] D. Or, E. Furtak-Cole, M. Berli, R. Shillito, H. Ebrahimian, H. Vahdat-Aboueshagh, and S. A. McKenna, "Review of wildfire modeling considering effects on land surfaces," *Earth-Science Reviews*, vol. 245, p. 104569, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012825223002581

[11] L. Šerić, J. Marasović, and D. Stipanicev, "Fire modeling in forest fire management," 01 2005.

[12] N. Burrows, "McArthur's Forest Fire Danger Meter and the Forest Fire Behaviour Tables for Western Australia: Derivation, applications and limitations," 07 1988.

[13] "Development and Structure of the Canadian Forest Fire Behavior Prediction System," 1992. [Online]. Available: https://cfs.nrcan.gc.ca/publications?id=10068

[14] S. J. Price and M. J. Germino, "Modeling of fire spread in sagebrush steppe using FARSITE: an approach to improving input data and simulation accuracy," *Fire Ecology*, vol. 18, no. 1, p. 23, 2022. [Online]. Available: https://doi.org/10.1186/s42408-022-00147-2

[15] R. Rothermel, "A mathematical model for predicting fire spread in wildland fuels." *U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station*, 1971.

[16] M. A. Finney, *FARSITE: Fire Area Simulator-model development and evaluation*, 1998. [Online]. Available: http://dx.doi.org/10.2737/RMRS-RP-4

[17] C. Tymstra, R. Bryce, B. Wotton, S. Taylor, and O. Armitage, "Development and structure of Prometheus: the Canadian Wildland Fire Growth Simulation Model," Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre, Edmonton, Alberta, Information Report NOR-X-417, 2010.

[18] J. Coen, "Simulation of the Big Elk Fire using coupled atmosphere-fire modeling," *International Journal of Wildland Fire*, vol. 14, 01 2005.

[19] W. Mell, M. Jenkins, J. Gould, and P. Cheney, "A physics-based approach to modeling grassland fires," *International Journal of Wildland Fire - INT J WILDLAND FIRE*, vol. 16, 01 2007.

[20] W. Mell, D. McNamara, A. Maranghides, R. Mcdermott, G. Forney, C. Hoffman, and M. Ginder, "Computer modelling of wildland-urban interface fires," *Conference Proceedings - Fire and Materials 2011, 12th International Conference and Exhibition*, 01 2011.

[21] J. J. Kari, *Basic Concepts of Cellular Automata*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–24. [Online]. Available: https://doi.org/10.1007/978-3-540-92910-9_1

[22] B. Drossel and F. Schwabl, "Self-organized critical forest-fire model," *Phys. Rev. Lett.*, vol. 69, pp. 1629–1632, Sep 1992. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.69.1629

[23] G. Efstathiou, S. Gkantonas, A. Giusti, E. Mastorakos, C. M. Foale, and R. R. Foale, "Simulation of the December 2021 Marshall fire with a hybrid stochastic Lagrangian-cellular automata model," *Fire Safety Journal*, vol. 138, p. 103795, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0379711223000632

[24] J. G. Freire and C. C. DaCamara, "Using cellular automata to simulate wildfire propagation and to assist in fire management," *Natural Hazards and Earth System Sciences*, vol. 19, no. 1, pp. 169–179, 2019. [Online]. Available: https://nhess.copernicus.org/articles/19/169/2019/

[25] A. Trucchia, M. D'Andrea, F. Baghino, P. Fiorucci, L. Ferraris, D. Negro, A. Gollini, and M. Severino, "PROPAGATOR: An Operational Cellular-Automata Based Wildfire Simulator," *Fire*, vol. 3, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2571-6255/3/3/26

[26] Y. Xu, D. Li, H. Ma, R. Lin, and F. Zhang, "Modeling Forest Fire Spread Using Machine Learning-Based Cellular Automata in a GIS Environment," *Forests*, vol. 13, no. 12, 2022. [Online]. Available: https://www.mdpi.com/1999-4907/13/12/1974

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[29] E. Mastorakos, S. Gkantonas, G. Efstathiou, and A. Giusti, "A hybrid stochastic Lagrangian – cellular automata framework for modelling fire propagation in inhomogeneous terrains," *Proceedings of the Combustion Institute*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1540748922002838

[30] S. B. Pope, "Lagrangian PDF Methods for Turbulent Flows," *Annual Review of Fluid Mechanics*, vol. 26, no. 1, pp. 23–63, 1994. [Online]. Available: https://doi.org/10.1146/annurev.fl.26.010194.000323

[31] "Copernicus Land Monitoring Service," https://land.copernicus.eu/en, accessed: 25.12.2023.

[32] "CLC+ Backbone," https://land.copernicus.eu/en/products/clc-backbone, accessed: 25.12.2023.

[33] "ChatGPT," https://openai.com/chatgpt, accessed: 24.01.2024.