1 Introduction

Forest fires are a natural and often necessary occurrence in many ecosystems. However, climate change is increasing the frequency and severity of these fires. As the planet warms, forests are becoming drier and more prone to ignition, leading to more intense and destructive fires that can threaten human communities, wildlife, and critical infrastructure. In the future, it is likely that even in areas such as the Black Forest which aren't historically prone to wildfires, the frequency of forest fires will increasingly pose a threat to people and the environment, unless steps are taken to improve fire management strategies [1, 2].
Computational tools such as simulation can be used to predict the spread and impact of forest fires and guide decision-making for prevention and response efforts. These tools can also be used to test different fire management strategies and policies to identify effective approaches for reducing the risk and impact of forest fires. Various different approaches, such as machine learning algorithms or spatial modelling have shown to be effective in predicting forest fire risk or simulating fire propagation [3, 4, 5].

In this paper, I present a small scale 3D cellular automaton that simulates the propagation forest fires using LIDAR data from terrestrial laser scans. This implementation has the potential to provide valuable insights into the dynamics and behaviour of potential forest fires in specific stands, as well as potential applications in the management and prevention of such fires by accurately determining vulnerable areas.
The main advantage of using a 3D cellular automaton model for simulating forest fires is the ability to incorporate detailed spatial and temporal information about the forest environment. By using data from terrestrial laser scans, this model can accurately represent the terrain, vegetation, and other physical features of the forest, allowing for more high resolution and accurate simulations.

1.1 Background and previous research

A cellular automaton is a mathematical model that is used to simulate the behavior of systems. It is made up of a grid of cells, each of which can be in one of a finite number of states. The state of each cell is determined by a set of rules that depend on the state of the cell and its neighbours. At each time step, the state of each cell is updated based on the rules, and the process is repeated to simulate the evolution of the system over time. Cellular automata are often used to model processes in fields such as physics, biology, and computer science, and can be used to simulate a wide range of phenomena (e. g. Earthquakes, the spread of diseases or urban dynamics) [6, 7].
There is a significant amount of research on the use of cellular automata for simulating forest fires [5, 8, 9], as well as the use of LIDAR data for characterising forest environments [10].
In terms of cellular automata models for simulating forest fires, most of the existing research has focused on large scale 2D models. These models use a grid-like structure to represent large areas of forest, with each cell representing a specific location at resolutions usually depending on satellite image resolution often around 20m diameter per cell [11, 9].
The most well-known example of a 2D cellular automata model for simulating forest fires is the "Self-organized critical forest-fire model" proposed by Drossel and Schwabl in 1992 [12]. This model has been widely studied and has inspired numerous variations and extensions, including the development of more realistic fire behavior, the incorporation of different types of vegetation, and the implementation of fire suppression strategies. These studies all implemented a different subset of parameters which

were then used to define the rules for the state of each cell.
A recent study by Byari et al in 2022 [13] showcases the application a 3D cellular automata implemented to simulate forest fire propagation. This multi-scale approach aimed to model heat transfer in detail on a small-scale and the overall propagation on a larger scale, while utilising LIDAR as well as satellite imagery data. The resulting resolution of cells for this implementation was a diameter of 10m per cell. However, to accurately model the fire dynamics within a specific stand and identify vulnerable locations, a smaller scale approach is necessary using i. e. terrestrial laser scan data at a much higher resolution.

1.2 My approach

For this approach a 3D cellular automata was programmed within the R programming environment [14] utilising the Rstudio IDE. First the terrestrial laser scan data is processed, voxelized and the resulting point densities represented in a 3-dimensional array which is basis for the cellular automaton. The state of the voxels in the array can can either be burning or non-burning. Voxels are "ignited" at a set probability if any voxel within the neighbourhood has the state "burning". The neighbourhood is determined by the parameters wind, wind strength and ignition range.
In the following I will explain the functioning of the 3D cellular automata in detail and discuss some of its shortcomings and possibilities to improve upon the prototype.

2 Methods and Results

2.1 Data pre-proccessing

Using the TreeLs package [15] the laser scan data is loaded into R. It is normalised and utilising the CSF algorithm developed by Zhang et al. [16], the ground is classified and removed. To filter only burnable fuel within the point cloud, stems are also classified and removed before further processing. After this process all further steps until visualisation take place within the *voxel automata* function.
From the TreeLS package the function *voxel metrics* regularises the point cloud, to create equally sized voxels with their respective point densities. This process determines the resolution at which the simulation will run (Figure 1). The resulting data frame now contains X,Y and Z coordinates and corresponding point density values for each voxel. Further, a data frame containing all coordinates within the extent of the point cloud is created and merged with the voxel data to include all empty voxels within the range of the point cloud.

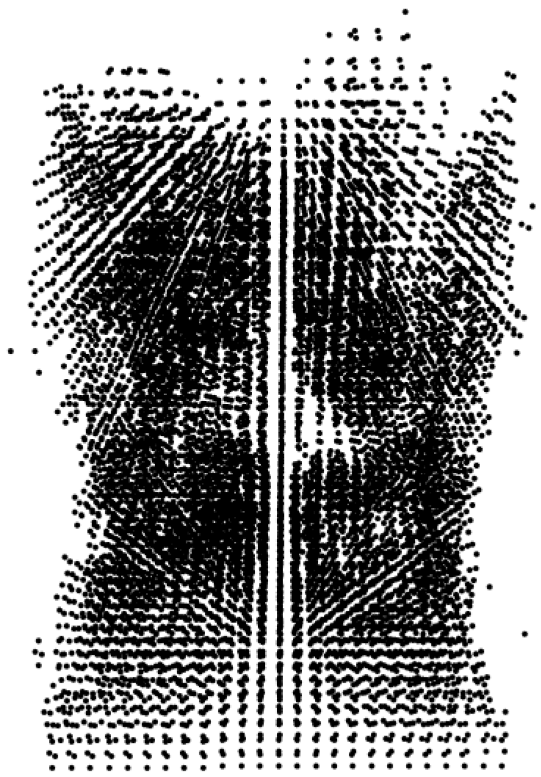

Figure 1: Example of a voxelized terrestrial laser scan with each point representing a voxel which exceeds a fill threshold

Using the following formula, the X, Y and Z coordinates are transformed to whole numbers and a minimum of 1:

$$X_{trans} = X * \frac{1}{Resolution} - min(X * \frac{1}{Resolution}) + 1$$

2.2 Determining the neighborhood

Table 1: All user-defined parameters for running the main function of the automaton.

Paramater	Description
res	the resolution at which the point cloud will be voxelized
start	the coordinates of the start of the fire
time	how many time-steps the simulation will run
fill_threshhold	the minimum amount of points within a voxel to be able to burn
range	the base size of the ignition neighbourhood
wind direction	the direction of wind
strength	the strength at which wind increase (or decreases) the ignition range
prob	the probability of a voxel igniting with a burning neighbour

To determine the size and shape of a neighborhood which can ignite a voxel the function *which neighbours* (Listing 1) takes into account the range, wind direction, wind strength and resolution (Table 1). A certain wind direction (N, NE, E, SE, S, SW, W or NW) will increase the range of ignition in the direction of wind and decrease it in the opposite direction. The amount increase and decrease depends on the defined wind strength. The result is a data-frame containing the coordinates of all voxels within this neighborhood expressed as a relative X, Y and Z distances to the a single voxel at (0,0,0).

```
which_neighboors <- function(range=1, wind_direction = F, strength=1, res){
  range = range / res
  strength = strength / res
  if(wind_direction == "F"){
    neighboorhood = expand.grid(x = -range:(range),
                                y = -range:range,
                                z = -range:range)
  } else {
    if(wind_direction== "N"){
      neighboorhood = expand.grid(x = -(range):range,
                                  y = -(range+strength):range,
                                  z = -range:range)
    }

    if(wind_direction == "E"){
      neighboorhood = expand.grid(x = -(range+strength):range,
                                  y = -range:range,
                                  z = -range:range)
    }
    .
    .
    .
  }
  return(neighboorhood)
}
```

Listing 1: R Code snippet for the "which neighbours" - function

2.3 Array buffer

With the ignition neighbourhood defined, a X, Y and Z buffer the size of the maximum neighbourhood distance is added to the data frame of coordinates. This is to prevent the later functions indexing voxels which are out of bounds. The chosen start coordinates of the fire are also transformed according to the buffer size. Now a three-dimensional array is created from the resulting dimensions over which the automaton will run. In this array each position will either be 0 or 1 (non-burning or burning).

2.4 Initiating array

The defined starting point in the array is given the value 1 which represents the state ”burning”. This is done by using the *‘foo¡-‘* function, which is used to assign values to a position in an array by indexing with a vector.
To log all coordinates of voxels which have been ignited, as well as at which time-step they were ignited, the data-frame “burnflag” is created.

2.5 Running the automaton

For each time-step in the loop of the automata three main processes take place:

(a) Checking the neighbourhood of all voxels for burning voxels.

(b) Transforming the array.

(c) Recording the burning voxel and the time-step of ignition

To determine if a voxel will be checked for burning neighbours, three conditions are in place which are checked for all voxels in each time-step: (1) a stochastic function with a user defined probability (Table 1), (2) a test if the voxel fill value exceeds the user defined threshold (Table 1) and (3) if the voxel is already burning. The latter is in place since burning voxels cannot extinguish and therefore do not need be checked again after they have burned. If a voxel full-fills these conditions the *check neighboors* (Listing 2) function is applied. This function returns a logical value depending on if any voxel within the calculated neighbourhood have the state burning.

```
check_neighboors = function(x, neighboorhood, array, coordinates){
  ft <- F
  for (k in 1:nrow(neighboorhood)) {
    if(array[coordinates$x[x]+neighboorhood$x[k],
             coordinates$y[x]+neighboorhood$y[k],
             coordinates$z[x]+neighboorhood$z[k]] == 1){
      ft <- T
      break()
    }
  }
  return(ft)
}
```

Listing 2: R Code snippet for the ”check neighbours” - function

If the *check neighboors* function yields true, then the coordinates of the respective voxel and the current time-step are added to the burnflag data frame. After iterating this through all voxels within the array, the voxels within the data frame burnflag, are given the state burning in the array using the *¡-foo* function.

2.6 Processing output

After all time steps have been completed, a data frame with the coordinates of all burned voxels and the time-step of ignition is created. Then using a join to the original data-frame of the voxels, the coordinates are back transformed to the geo-referenced coordinates. The return of the function *voxel* *automata* consists of a data-frame including all back-transformed voxel coordinates and their state after the automata has run, the resulting 3D array and the burnflag data frame.

2.7 Visualisation

For visualisation of the resulting point clouds the rgl package [17] can be used to create 3D interactive graphics and further inspect the results. This could then be used to identify critical and vulnerable locations regarding the spread of fire within the stand (Figure 2).

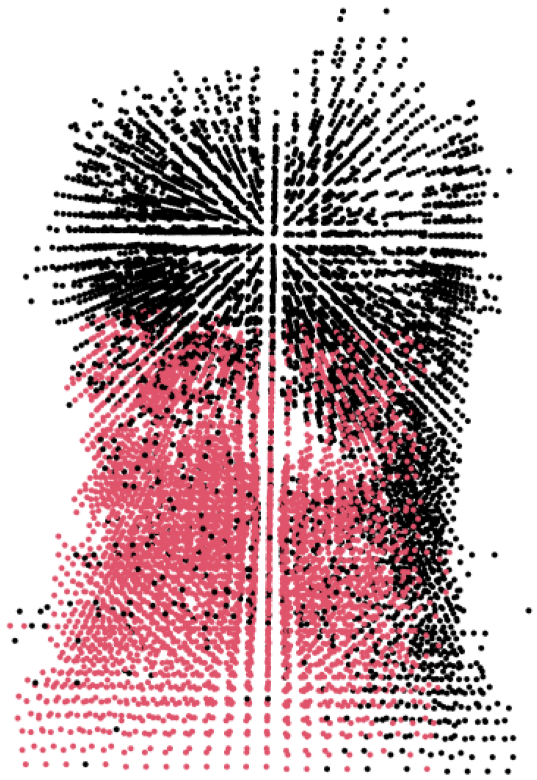

Figure 2: Resulting point cloud after a completed simulation (used parameters: *time* = 10, *range* = 2, *wind direction* = "N", *strength* = 1, *res* = 1, *prob* = 0.5, *fill threshold* = 30). Red points represent voxels which have burned, black points are unburnt voxels.

Table 2: Set of parameters used for evaluating runtime and the resulting mean runtime after

time	10	10
range	2	2
wind direction	”N”	”N”
strength	1	1
res	1	0.5
prob	0.5	0.5
fill threshold	30	30
Mean runtime	69.4 s	1409.2 s

3 Discussion

3.1 Performance

Regarding performance, the current version of the automaton is relatively inefficient from a computational standpoint and most likely unusable when applied to larger data-sets, more time-steps or a higher resolution. Currently the average run-time (macOS 10.15, 8GB RAM and 2.3 GHz Dual-Core Intel Core i5) with the set of parameters shown in Table 2 is increased by 20 times when doubling the resolution, illustrating the difficulty of scaling this model. Especially when aiming to extend the simulation by implementing more parameters a more efficient programming is necessary.
The largest factor concerning run-time are the nested loops in the checking and transforming part of the automaton function. Reducing the amount of loop-iterations would yield a significant improvement over the current run-time. The largest improvement of efficiency in this regard would be an inversion of the checking function. Instead of iterating through all voxels at every time-step and checking each neighborhood, only iterating through voxels which have the state burning and transforming their respective neighborhoods. This would especially in the first few time-steps drastically improve runtime as only very few voxels and consequently neighborhood have to be iterated through.

Further, instead of using for-loops in the main function, vectorized functions could yield a performance increase when utilised efficiently. This could be achieved by using functional programming toolkit such as *purrr* [18] or utilizing the *apply* functions from base R where possible. Alternatively this part of the function could also be implemented in a faster programming language such as C++ by using the *Rcpp* package which offers an integration of C++ in the R programming environment [19].

3.2 Parameters

3.2.1 Wind

Currently wind only decreases the spread of fire in direction a increases the spread in the opposite direction. This is a very simplistic representation of how wind effects fire in reality. An important aspect which could be implemented easily would be a reduction of the flame height (i.e. the neighborhood in the z-dimension) depending on the strength of wind. Another important improvement would be a more accurate options for the wind direction. Being able to to input a detailed angle would help to more accurately simulate the propagation of fire in a specific location. A much more complex improvement would also be including the interaction between the wind and the topography. Calculating new transition rules for every voxel depending on the interplay between the slope and wind.

3.2.2 Probability

The transition probability in this automaton represents the heat or intensity of the fire within the voxel. In this automaton this probability is static and the same for all voxels over every time-step. However as Encinas et al. (2007) [5] argue, the probability of voxel igniting other voxels should rather be dynamic for each burning voxel over time. As it grows as the material heats up and then consecutively decreases as the fuel within the voxel depletes. Adding this key feature of a dynamic likelihood of igniting would greatly improve the accuracy of the model. As this eliminates the problem of the automaton running over many time-steps and consequently "undermining" the ignition probability and eventually burning all voxels which are "physically connected" within the point cloud. To implement this the point density within each voxel could be used as a proxy for the fuel within the voxel which depletes over time and consequently reduces both the ignition probability and range.

3.3 Parametrisation

To correctly apply this automaton the used variables and transition rules would have to be correctly parameterized. This is traditionally done by the identification of parameters from experimental results [5, 3]. Predominantly information on the dynamics and speed of the spread and intensity of fires has been measured and applied to transition rules of these larger scale automatons. However, as Colin et al. (2011) [20] for the accurate representation of the much smaller scale dynamics of forest fires physical models containing the main physical processes involved in fire propagation - convection, radiative transfer, and pyrolysis degradation—to define the transition rules of the automaton.

3.4 Additional features

The presented automaton is a simple prototype and will need more adaptions, specifications and input variables to be able to accurately or realistically represent the dynamics of forest fires. The following section provides a short outlook, some ideas and recommendations for the expansion of this model.

(1) As the slope of the area heavily effects the spread behaviour of a forest fire, this should also be included in the transition rules of the automaton [21]. As the morphology of the area is included in the laser scan data this implementation would be rather simple.

(2) The inclusion of other climatic data, such as temperature and especially precipitation or rather the resulting humidity of the material in the area in which the automaton is applied, would likely greatly improve the accuracy of results, as the moisture of the material strongly impacts the fire strength and speed of spreading [20, 5].

(3) The classification of fuel-types from the LIDAR data to implement varying burning characteristics for each fuel type would further increase the accuracy of this model, as different fuel types will ignite and propagate the fire much differently [22]. However, the classification of fuel would have to occur at a very high resolution to improve this model. Previous research regarding this topic has mostly mapped fuel types at much larger scales by, for example, using the percentage of shrubs, trees and grass to categorize fuel types [23, 24]. If possible, an algorithm or model which classifies different

fuel types (e.g woody material, grass, leaves, or etc.) at a high resolution would be very beneficial to this implementation.

4 Conclusion

This paper presented a prototype of a 3D cellular automaton which uses LIDAR data from terrestrial laser scans and simulates the propagation of forest fires at a small scale to identify potential risk zones. Although functional, this implementation still needs many more adjustments and expansions to accurately predict the propagation of forest fires. Further parameters have to added and the existing ones parameterized. Further, for a model this complex to be feasible, a more efficient programming for better performance is necessary as it would allow much larger areas and a higher resolution.
However, models of this design could prove very useful in the future, especially in light of the advent of lightweight and affordable LIDAR scanners and long-range UAVs. Being able to fly over large areas of forest, run a cellular automaton and receive all potential risk zones and consequently apply this knowledge to forest management and prevent or mitigate potential fire damages could be a very important and beneficial tool in the future. Especially with the projected climate change increasing the frequency of forest fires efficient and strategic management is a requisite and computational tools such as this automaton will possibly play an important role.

References

[1] Phillip J van Mantgem, Jonathan CB Nesmith, MaryBeth Keifer, Eric E Knapp, Alan Flint, and Lorriane Flint. Climatic stress increases forest fire severity across the western u nited s tates. *Ecology letters*, 16(9):1151–1156, 2013.

[2] Fantina Tedim, Gavriil Xanthopoulos, and Vittorio Leone. Forest fires in europe: Facts and challenges. In *Wildfire hazards, risks and disasters*, pages 77–99. Elsevier, 2015.

[3] Mauro Castelli, Leonardo Vanneschi, and Aleš Popovič. Predicting burned areas of forest fires: an artificial intelligence approach. *Fire ecology*, 11(1):106–118, 2015.

[4] Binh Thai Pham, Abolfazl Jaafari, Mohammadtaghi Avand, Nadhir Al-Ansari, Tran Dinh Du, Hoang Phan Hai Yen, Tran Van Phong, Duy Huu Nguyen, Hiep Van Le, Davood Mafi-Gholami, et al. Performance evaluation of machine learning methods for forest fire modeling and prediction. *Symmetry*, 12(6):1022, 2020.

[5] L Hernández Encinas, S Hoya White, A Martín Del Rey, and G Rodríguez Sánchez. Modelling forest fire spread using hexagonal cellular automata. *Applied mathematical modelling*, 31(6):1213–1227, 2007.

[6] Kendall Preston Jr and Michael JB Duff. *Modern cellular automata: theory and applications*. Springer Science & Business Media, 2013.

[7] Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419–424, 1984.

[8] Ioannis Karafyllidis and Adonios Thanailakis. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling*, 99(1):87–97, 1997.

[9] S Yassemi, S Dragićević, and M Schmidt. Design and implementation of an integrated gis-based cellular automata model to characterize forest fire behaviour. *ecological modelling*, 210(1-2):71–84, 2008.

[10] Michael A Wulder, Joanne C White, Ross F Nelson, Erik Næsset, Hans Ole Ørka, Nicholas C Coops, Thomas Hilker, Christopher W Bater, and Terje Gobakken. Lidar sampling for large-area forest characterization: A review. *Remote sensing of environment*, 121:196–209, 2012.

[11] O Jellouli, A Bernoussi, M Mâatouk, and M Amharref. Forest fire modelling using cellular automata: application to the watershed oued laou (morocco). *Mathematical and computer modelling of dynamical systems*, 22(5):493–507, 2016.

[12] Self-organized critical forest-fire model. *Physical review letters*, 69(11):1629, 1992.

[13] M Byari, A Bernoussi, O Jellouli, M Ouardouz, and M Amharref. Multi-scale 3d cellular automata modeling: Application to wildland fire spread. *Chaos, Solitons & Fractals*, 164:112653, 2022.

[14] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2022.

[15] Tiago de Conto. *TreeLS: Terrestrial Point Cloud Processing of Forest Data*, 2022. R package version 2.0.5.

[16] Wuming Zhang, Jianbo Qi, Peng Wan, Hongtao Wang, Donghui Xie, Xiaoyan Wang, and Guangjian Yan. An easy-to-use airborne lidar data filtering method based on cloth simulation. *Remote sensing*, 8(6):501, 2016.

[17] Daniel Adler, Maintainer Duncan Murdoch, MASS Suggests, PLY WebGL, STL OBJ, and SystemRequirements OpenGL. Package 'rgl', 2019.

[18] Hadley Wickham and Lionel Henry. *purrr: Functional Programming Tools*, 2023. R package version 1.0.1.

[19] Dirk Eddelbuettel and James Joseph Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *The American Statistician*, 72(1):28–36, 2018.

[20] A Collin, D Bernardin, and O Sero-Guillaume. A physical-based cellular automaton model for forest-fire propagation. *Combustion science and technology*, 183(4):347–369, 2011.

[21] Domingos Xavier Viegas. Forest fire propagation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1748):2907–2928, 1998.

[22] WJ De Groot, JM Pritchard, and TJ Lynham. Forest floor fuel consumption and carbon emissions in canadian boreal forest fires. *Canadian Journal of Forest Research*, 39(2):367–382, 2009.

[23] Darío Domingo, Juan de la Riva, María Teresa Lamelas, Alberto García-Martín, Paloma Ibarra, Maite Echeverría, and Raúl Hoffrén. Fuel type classification using airborne laser scanning and sentinel 2 data in mediterranean forest affected by wildfires. *Remote Sensing*, 12(21):3660, 2020.

[24] Mariano García, David Riaño, Emilio Chuvieco, Javier Salas, and F Mark Danson. Multispectral and lidar data fusion for fuel type mapping using support vector machine and decision rules. *Remote Sensing of Environment*, 115(6):1369–1379, 2011.