Journal of
open research software

Abouali, A and Viegas, D X 2019 Fire ROS Calculator: A Tool to Measure the Rate of Spread of a Propagating Wildfire in a Laboratory Setting. *Journal of Open Research Software,* 7: 24. DOI: https://doi.org/10.5334/jors.221

SOFTWARE METAPAPER

Fire ROS Calculator: A Tool to Measure the Rate of Spread of a Propagating Wildfire in a Laboratory Setting

Abdelrahman Abouali and Domingos Xavier Viegas

ADAI/CEIF (Association for the development of Industrial Aerodynamics/Centre of Studies About Forest Fires), University of Coimbra, PT

Corresponding author: Abdelrahman Abouali (awabuali@hotmail.com)

The Fire ROS Calculator is a software tool built using MATLAB to assist researchers from the wildfire research area in analysing wildland fire behaviours. In particular, it measures the rate of spread (ROS) of a fire propagating over a surface in a laboratory setting and constructs its propagation contour map. An algorithm is used to calibrate the camera used for filming the fire spread. Various algorithms and image processing procedures are applied to the images to obtain the ROS. The software has a graphical user interface (GUI) and documentation stored in a GitHub repository alongside the source code.

Keywords: Rate of spread; Wildfire; Forest Fires; Fire Behaviour; camera calibration; Image Processing
Funding statement: The software resulted from the funding of the Portuguese Science Foundation for the project "FIREWHIRL – Vorticity Effects in Forest Fires" (PTDC/EMS-ENE/2530/2014).

(1) Overview

Introduction

The rate of spread (ROS) of a fire is one of the main parameters that describe wildland fire behavior, and it is commonly used in research and operational. We developed the Fire ROS Calculator software to assist researchers in measuring the fire's ROS on a laboratory scale with sufficient accuracy. On this scale, it is important to realise the small changes and transitions on the fire behavior as it might have significant consequences on field scales. The need for this software arises as there is no published software or program that the authors are aware of which is dedicated for this purpose. Most of the researchers from this area are using locally developed macros or programs to analyse their experiments. Also, often more than one software tool is used to obtain final outputs such as the fire's propagation contour map, which makes the analysis process slow and may produce low accuracy results. Fire ROS Calculator solves these problems by providing several automated tools to obtain accurate and customised outputs.

There are two main programs in the Fire ROS Calculator. The first is the calibration program; it performs a camera calibration process using functions from MATLAB's Computer Vision Toolbox. The calibration algorithm implements the camera model proposed by Jean-Yves Bouguet [1]. The model includes the Pinhole camera model [2] and a lens distortion model [3]. The algorithm uses a calibration object as a reference, which is a checkerboard, to calculate the camera extrinsic and intrinsic parameters (calibration matrix) [1]. There are detailed instructions about the calibration process on the software's documentation, which are saved in the GitHub repository.

The second main program does the detection of the fire front's location on a sequence of images captured from the same location. Then it converts the detected locations (pixel coordinates) to real-world coordinates using the produced calibration matrix. This process can obtain the distances that the fire has travelled over the course of time and thus the fire's ROS. The location of the fire front on the images can be detected manually by the user or automatically by a program. The automatic detection program uses an image-processing algorithm that is built using functions from MATLAB's Image Processing Toolbox. Finally, the software has several tools that allow the user to perform the whole analysis process starting from a video or images of the fire and ending with several commonly-used outputs like dynamical or averaged fire's ROS and the propagation contour map. The automatic detection can also detect several fires and track the evolution of their fronts with the advance in time.

Implementation and architecture

The software tool is built from several programs, where we will discuss the architecture of some of them. First, we are presenting the software's GUI. The GUI is constructed

from five tabs (**Figure 1**); each tab has a specific objective. The five tabs are.

New Session

The user can run from this tab a program that determines the locations of the fire front from inputted images and transforms them into polylines with real-world two-dimensional coordinates (X–Y). The fire front lines (polylines) from the different time steps are constructing together a propagation contour map where the real displacement of the fire with the time is defined. After the program finish, the user can obtain the outputs (like measuring the ROS along a direction defined by the user).

Load Session

From this tab, the user may load an old saved session (i.e., a file that contains information about the fire front locations and other parameters) to obtain different outputs.

Match Images

This tab contains a program that can resize images to match another input image. The tool is needed when two different cameras are used in the calibration process. The software's manual has an explanation of when it could be required to use two cameras for calibration.

Extract Frames

This tab has a tool that enables the user to extract frames from a video with a defined time-lapse between them.

Camera Calibration

The tab contains a calibration tool, which obtains the Pinhole camera parameters [2] and saves them in a file to be used later in analysing any fire in which the same camera was used to film it.

We will present the architecture of two programs; first is the program that detects the locations of the fire fronts on the images and transforms them to real-world dimensions, the other is the program that calculates the fire's ROS. To keep the article focused, we do not present here the architecture of the rest of the programs, as these two are the most important ones in the software. Regarding the program that performs the camera calibration, we are not presenting it as well since it has a flow similar to the one presented on the MATLAB tutorial for single camera calibration [5]. We show the two programs, the fire front detector and the ROS calculator, in the form of flowcharts in **Figures 2** and **3** with briefings about them on the following.

The fire front Detector

The program runs only after the user fulfilled all the inputs on the "New Session" tab (**Figure 1**) which are presented below:

- The Camera Parameters file, the file can be generated for a camera using the Camera Calibration Tool on the Camera Calibration tab.
- The fire images (frames), which show the fire's propagation.

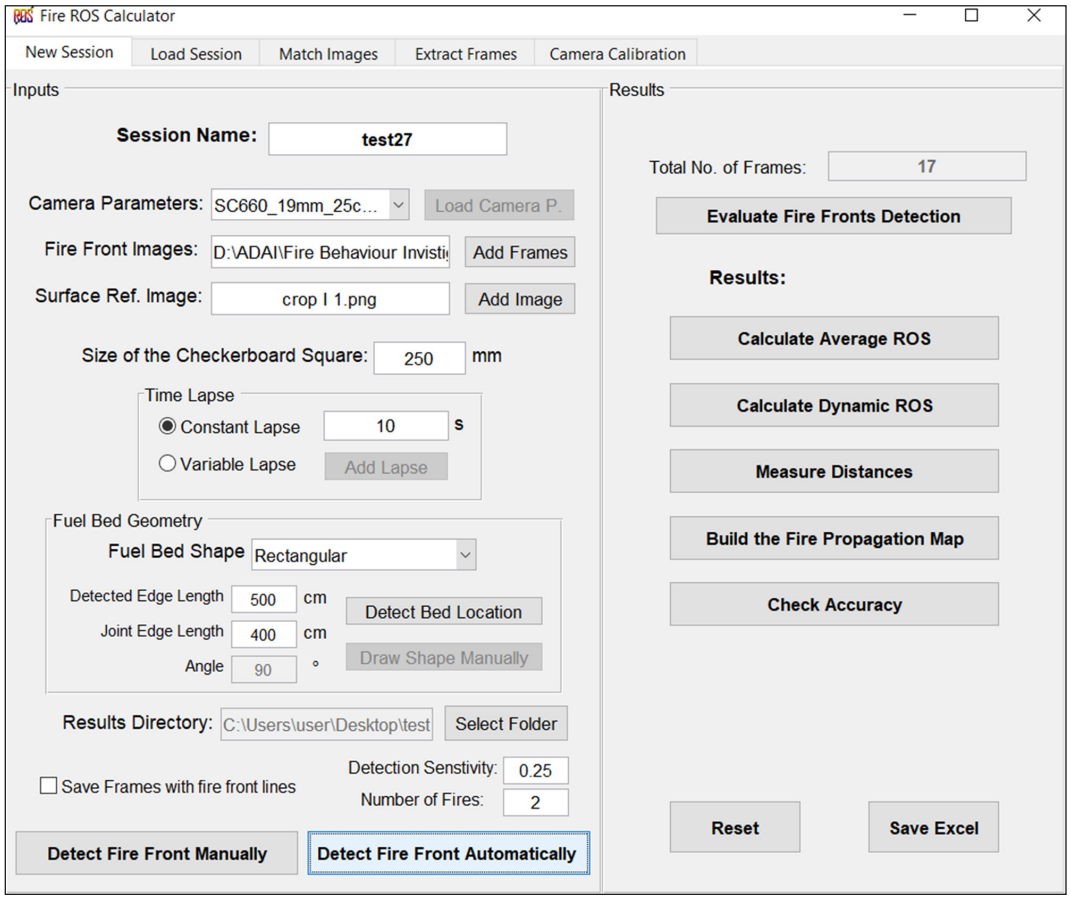

Figure 1: An image showing the graphical user interface of the Fire ROS Calculator where the activated tab is the "New Session" tab.

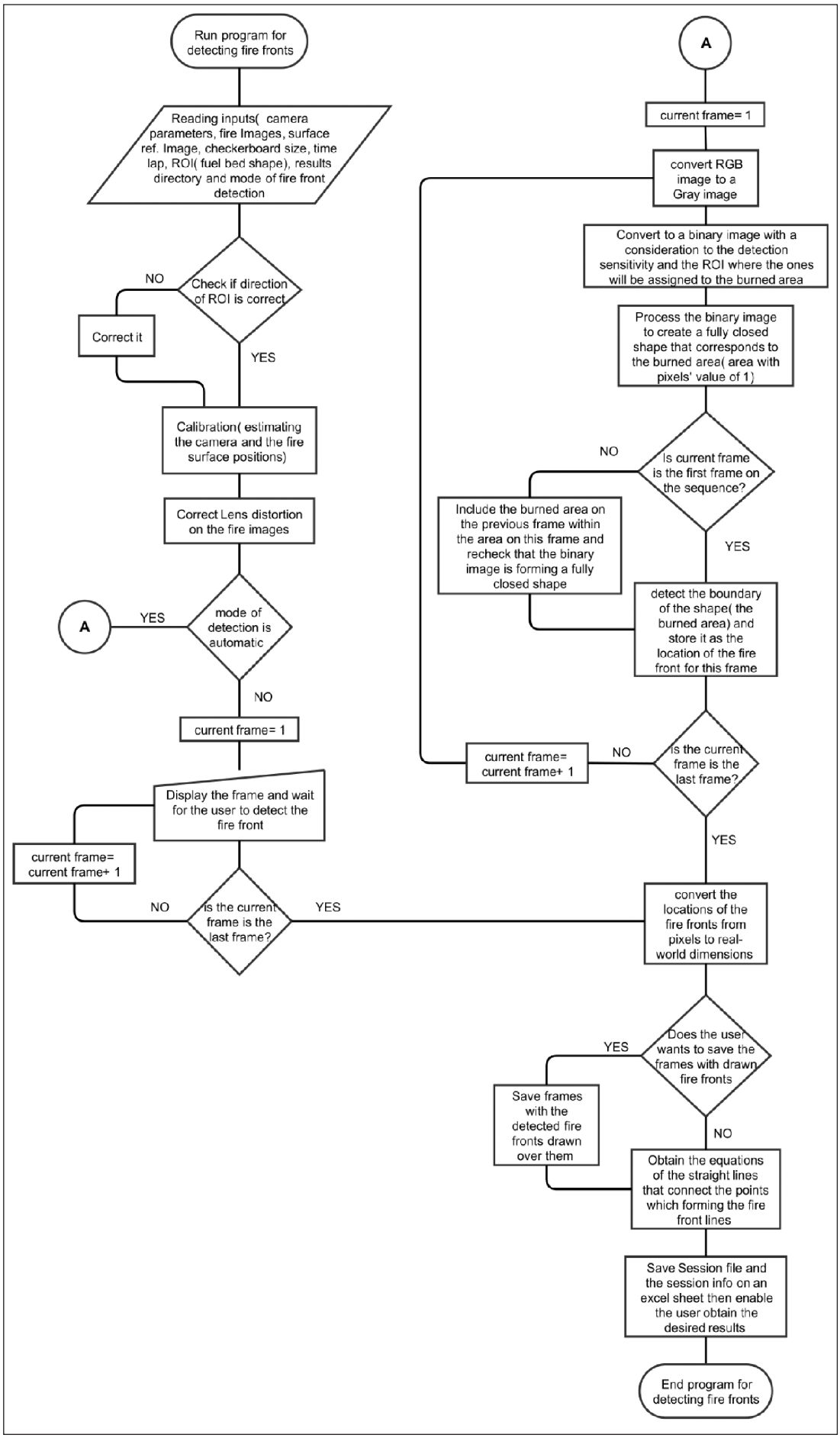

Figure 2: The flow chart of the program that detects the fire front pixel- coordinates on the images and convert them to real-world coordinates.

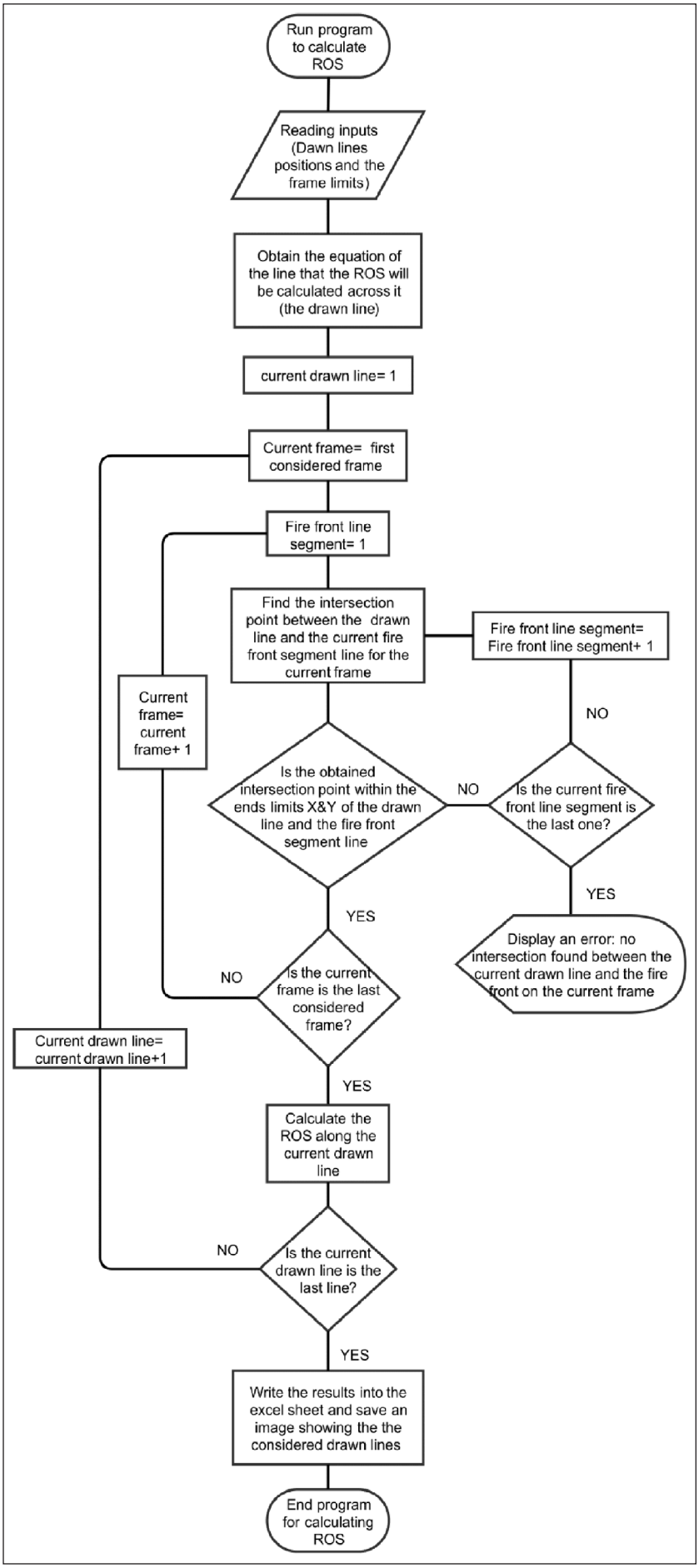

Figure 3: The flow chart of the program that calculates the ROS of the fire.

- The Surface Ref. Image. An image captured for the calibration object placed over the fire's propagation surface. The image is used to determine the camera position relative to the surface. There is a detailed explanation of this image on the software's documentation.
- The size of the checkerboard square, which is a property of the calibration object.
- The time-lapse between the frames.
- Fuel bed shape, which is the ROI (region of interest) for the image processing algorithms. The user can determine it as a regular shape (rectangular or triangle) or draw the shape manually.
- The results directory location and selecting wither the user wants to save or not the fire images with the obtained fire front lines drawn over them.
- The detection sensitivity in a case the user will use the automatic detection mode and the number of fires that the program should consider.

The program runs after the user clicks over one of the detection modes, "Detect Fire Front Manually" or "Detect Fire Front Automatically" (**Figure 1**). The program starts with reading the inputs and then will do a calibration process to determine the relative position of the camera. In the case of the automatic detection mode, the algorithm is designed to analyse IR images where it performs a set of image processing processes for each frame. The algorithm can work with other types of images, but accuracy is not guaranteed.

The process starts by converting the image to grayscale then to a binary image where it assigns a value of one (white) to the pixels that correspond to the burned area on the image, and zero (black) to the remaining the rest of the pixels. The program determines the pixels as burned or not based on its brightness and considers the historical locations of the fire front (i.e. on the previous frames) in an additive process, this leads to adding the detected bright pixels on a frame to the previously detected pixels on the previous frames to define the burned area. The selected detection sensitivity controls the refinement of this detection or the degree of the pixel's brightness that should be included within the burned area. Then the algorithm processes the binary image to ensure that the detected burned areas are forming a fully closed and connected area. Finally, on the last process, it obtains the coordinates of the fire front line (i.e. the perimeter of the detected burned area) (**Figure 4**), which is formed by a polyline.

After obtaining the fire fronts' pixel coordinates, the program converts them to real-world coordinates. After finishing, the program saves a file that contains calibration information and the detected fire front location along with other data. This file can be loaded to the software later in another session to get any output from the "Load Session" tab.

The ROS Calculator

The software provides two options to calculate a ROS; one is to calculate the Averaged ROS along a predefined direction by the user, and the other is to calculate a Dynamic ROS. The Averaged ROS describes the fire's spread

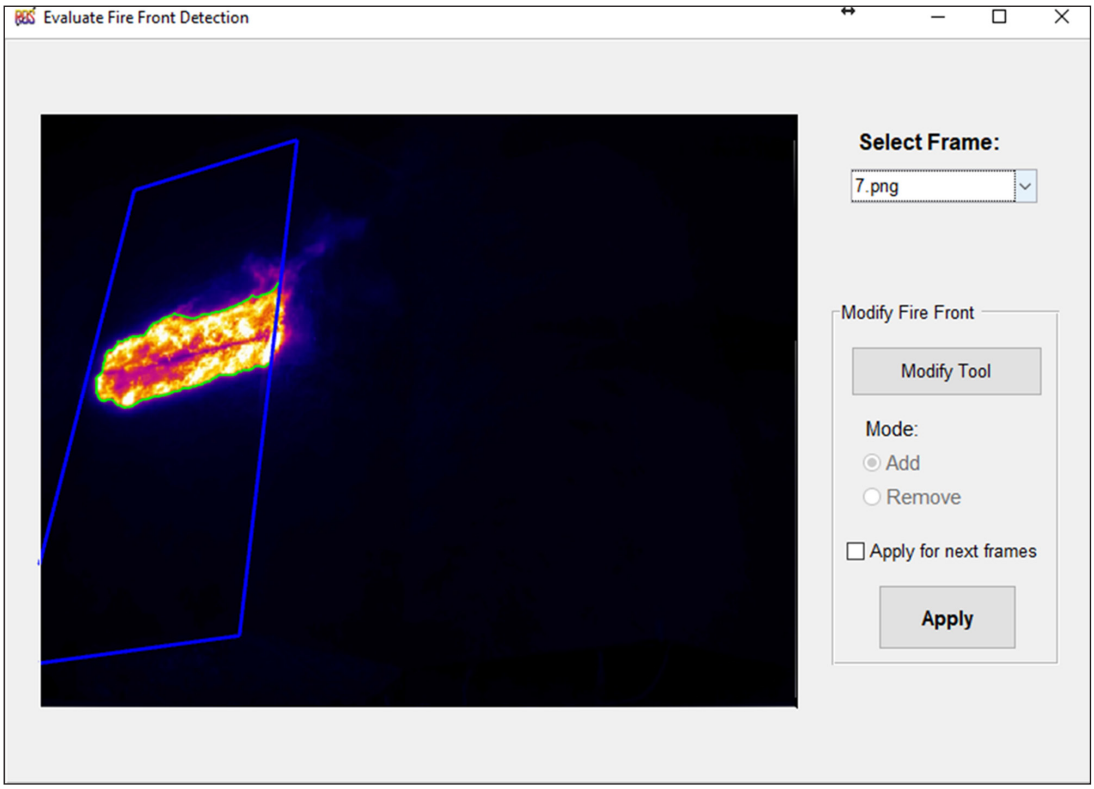

Figure 4: Image showing the window where the user can evaluate the automatic detection of the fire front. On the image, the green line is the detected fire front on an IR image and the blue line is the detection of the surface of the fire propagation (the fuel bed).

rate in a given direction averaged over time. The Dynamic ROS describes the fire's spread rate dynamically with the time, which means the ROS that the fire has travelled with between every two frames along the defined direction by the user. Both programs to calculate averaged and dynamic ROS have similar architecture. The Averaged ROS program has a feature that allows calculating the ROS over several lines (directions) and then calculate the average of all these lines. This feature can be used to better describe the ROS of the fire in some direction.

Two inputs are needed to run the program; first is the direction where the ROS will be calculated, which is input as a line that the user draws over the fire's propagation contour map (**Figure 5**). The second input is the frame limits, which are the considered frames to calculate the ROS between them. Defining the limits gives the user the flexibility to calculate the ROS within some exact time steps.

To measure the ROS, the program does the following operations:

- Starts by reading the inputs, which are the frame limits (first and last frame) and coordinates of the direction-line (the one drawn by the user).
- Then obtains the equation of the direction-line.
- An algorithm scans the fire front polyline on each frame to find an intersection with the direction line(s).
- After defining the intersections between the direction-line and the fire front on each frame, the program calculates the distances between the intersections along the direction-line.
- If the program did not find an intersection between the direction-line and the fire front's polyline on a frame, the program displays an error with the order of the frame that the algorithm could not find an interaction with it. This error usually happens if the user did not place the line properly to intersect will all the frames within the determined frames' range.
- After calculating the distances, the program calculates the ROS using the given time intervals between the frames.

We define the average rate of spread (ROS) of a fire along a prescribed direction as the slope of a linear fit between the two data sets, the distances and their associated times, following Viegas (2004) [4]. However, this simplification to calculate the average ROS is only acceptable if there are no consistent variations of the rate of spread (ROS) and it implies that the fire is spreading in a quasi-steady-state, which may not be valid in all cases [4].

Finally, the program writes the results to an Excel sheet and saves an image showing the location of the direction lines where ROS was calculated.

Quality control

We tested the program and validated its results by performing three experiments in the Forest Fire Research Laboratory of the University of Coimbra in Lousã (Portugal). The output value of the Averaged ROS of a propagating fire was compared to a reference measurement of the ROS taken for the same experiment; then we calculated the errors. Validating this output quantity, the average ROS, can

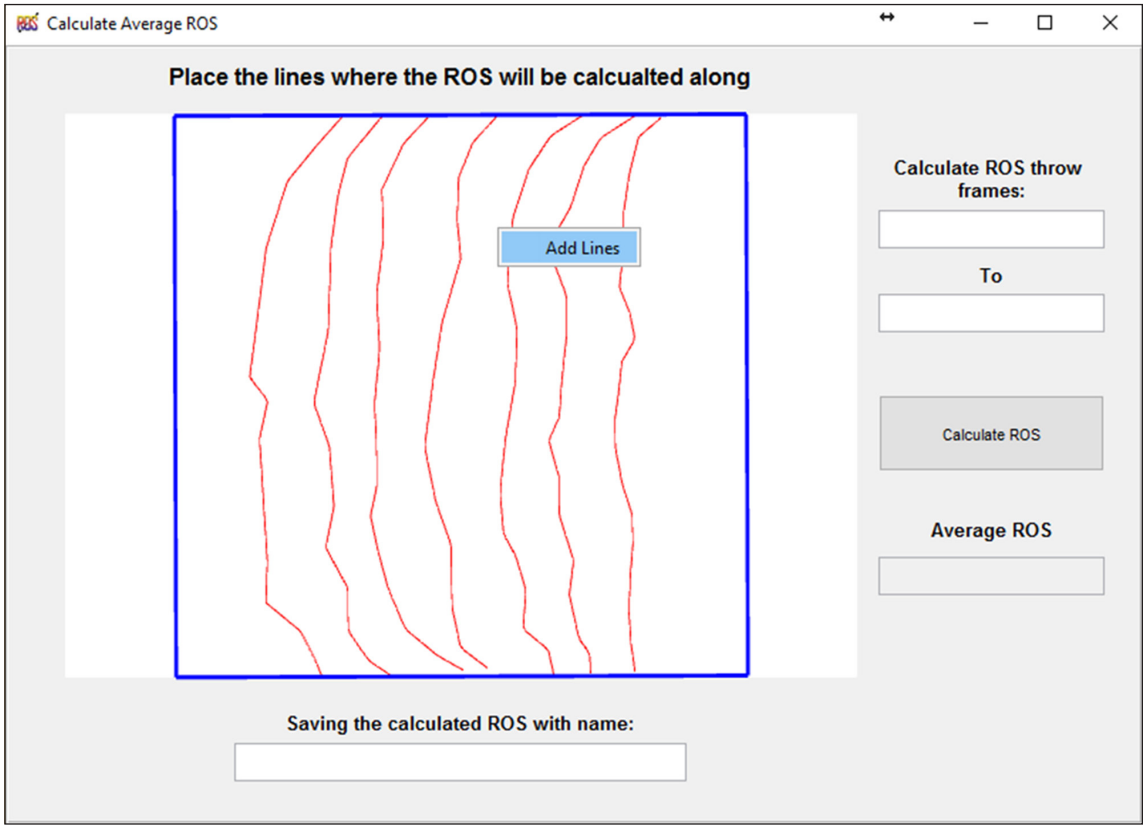

Figure 5: Image showing the window of the program that calculates the average ROS.

be considered a validation for the software as a whole as it uses all the major programs to calculate the average ROS. The experiments were performed in a no-wind condition over a square flat combustion-table with an area of 1 m^2. The table was prepared with a uniform fuel bed composed of dead pine needles (*Pinus Pinaster*) with a load of 0.6 kg.m-2 (dry basis) and a depth of 0.05 ± 0.01 m. A surface fire was originated from a line along one of the table's edges, and we recorded the fire with an IR camera. The used camera has the model FLIR SC660 with a resolution of 640 × 480. During the tests, we take a reference measurement for the fire's ROS where the testing table was prepared with a set of lines (strings) parallel to the ignition line and spaced by 10 cm from each other. We recorded the time taken by the fire front to travel from a line to another. The average ROS was calculated by fitting a linear regression between the times and their associated distances, then calculating the slope of the line. This methodology is the same methodology that the program is using to calculate the average ROS.

The program output, the average ROS was obtained be acquiring several frames from the IR film of the fire with a time-lapse of 30 seconds between them. We calibrated the camera following the same mentioned procedure in the Fire ROS Calculator's Manual. The three tests had different calibrations and camera positions, which we present in **Table 1**; this was considered to ensure the quality of the validation.

In this combination of tests, we used two different camera lenses, one with wide-angle zoom and another with a higher zoom (19 mm and 42 mm). Also, two calibration objects of various sizes were used. We used the Manual fire front detection option for all tests except test C, where we analysed it using both Manual and Automatic detection options. The average ROS of the fire was calculated using the "Calculate Average ROS" tool in the software. We report the results in **Table 2** along with the percentage of residual errors in comparison with the reference measurements. We conclude from the reported results that the outputs of the software have a error margin of ±5%.

The inputs of Test C is available on the GitHub repository in a compressed file (Case C Materials.rar), where the user may run the case and calculate the output ROS to validate it with the reported reference measurement. There is a brief explanation on how to run the case on the Fire ROS Calculator's Manual on the Validation and Testing section.

(2) Availability

Operating system

The available compiled version is for Windows; the user may ask the authors for a compiled version for other OSs. Also, if the user has MATLAB, they can run the Fire ROS Calculator directly as a MATLAB function.

Programming language

MATLAB, minimum required MATLAB 2018a.

Additional system requirements

Preferred requirements: Memory of 4 Gb and Disk Free Space of 2 Gb. The software can run with lower system resources, but it will be slow.

Dependencies

The MATLAB Runtime R2018a Libraries set is needed to run the compiled version. The installer will download it from the internet and install it automatically if needed.

Software location

Archive

Name: Zenodo

Table 1: The performed tests' camera and calibration parameters.

Test	Camera's altitude from the table	Used lens	Calibration Object
A	8 m	42 mm lens	2 × 1 m with 25 cm square size
B	4 m	19 mm lens	2 × 1 m with 25 cm square size
C	4 m	19 mm lens	0.45 × 0.9 m with 15 cm square size

Table 2: The average fire ROS results obtained from the software and the reference measurement for each test are reported along with the residual errors.

Test	Test A		Test B		Test C		Test C (Automatic Detection)	
Reference ROS (mm/s)	**3.15**		**3.61**		**3.64**		**3.64**	
Quantity	**ROS (mm/s)**	**Error (%)**	**ROS (mm/s)**	**Error (%)**	**ROS (mm/s)**	**Error (%)**	**ROS (mm/s)**	**Error (%)**
Line 1	3.2	2.9	3.5	2.2	3.6	0.5	3.6	1.9
Line 2	3.3	4.4	3.6	0.3	3.5	4.4	3.5	4.9
Line 3	3.2	2.9	3.8	4.4	3.5	4.7	3.6	1.7
Average	3.3	3.4	3.6	0.6	3.5	3.2	3.5	2.8

Persistent identifier: DOI: 10.5281/zenodo.3252519
Licence: *GNU General Public License V3*
Publisher: Abdelrahman Abouali
Version published: 2.6
Date published: 22/06/2019

Code repository

Name: GitHub
Identifier: github.com/AAbouali/Fire_ROS_Calculator
Licence: *GNU General Public License V3*
Date published: 14/03/2017

Language

English

(3) Reuse potential

The software is mainly dedicated to researchers from the wildfire area, particularly, the researcher in which their work involves experimental laboratory testing for fire behaviors and propagation. The primary objective of the software is to measure the fire's ROS and construct its propagation contour map. However, it can be used for other objectives whenever the user aims to know the real-world dimensions of an object from an image or track that object movement over a surface and calculate its velocity using the Manual detection. However, with some modification to the image processing algorithm, the detection of the object in each time step (frame) can be made automatically as well. This general objective can extend the potential of use to other applications that require converting image pixel coordinates to real-world coordinates.

The authors are planning to extend the use of this software to include also measuring the ROS of a fire propagating in an outdoor setting, and they are welcoming any contributions to this software especially if it will include more analysis capabilities for the wildland fire researchers.

For support, please contact the authors via the email address given on the ORCID page: (orcid.org/0000-0002-1839-5149).

Acknowledgements

The Authors would like to acknowledge the contributions of ADAI's team, namely Jorge Rafael Raposo and Nuno Luís for their contribution to verify the results of the software.

Competing Interests

The authors have no competing interests to declare.

References

1. **Bouguet, J Y** "Camera Calibration Toolbox for Matlab." Computational Vision at the California Institute of Technology.
2. **Zhang, Z** 2000 A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11): 1330–1334. DOI: https://doi.org/10.1109/34.888718
3. **Heikkila, J** and **Silven, O** 1997 A Four-step Camera Calibration Procedure with Implicit Image Correction. *IEEE International Conference on Computer Vision and Pattern Recognition.*
4. **Viegas, D X** 2004 Slope and Wind Effects on Fire Propagation. *International Journal of Wildland Fire*, 13(2): 143–56. DOI: https://doi.org/10.1071/WF03046
5. MATLAB Documentations: Computer Vision System Toolbox, Single Camera Calibration. https://www.mathworks.com/help/vision/single-camera-calibration.html.

How to cite this article: Abouali, A and Viegas, D X 2019 Fire ROS Calculator: A Tool to Measure the Rate of Spread of a Propagating Wildfire in a Laboratory Setting. *Journal of Open Research Software,* 7: 24. DOI: https://doi.org/10.5334/jors.221

Submitted: 09 February 2018 **Accepted:** 18 July 2019 **Published:** 29 July 2019

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS