deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

University of Aveiro

FIRELAB: Wildfire Imagery Analysis and Processing

António Fernandes
David Morais
Francisca Barros
Luís Silva
Mariana Ladeiro

Technical Report
First Cycle Degree in Informatics Engineering
Supervisors: José Moreira, Hélder Zagalo, Patrícia Fernandes
25th June 2021

This essay portrays the final project developed for the curricular unit of Project in Informatics, included in the First Cycle Degree in Informatics Engineering (LEI) at the University of Aveiro (UA). The project was carried out by Antonio Fernandes, David Morais, Francisca Barros, Luís Silva and Mariana Ladeiro, under the guidance of Professor José Moreira, Assistant Professor at DETI, Professor Assistant Professor at DETI and Ana Patricia Fernandes, Assistant Researcher at DAO.
The work focuses on implementing a set of tools to transform images and videos (raw data) in reliable georeferenced data, ready to be used by investigators, environmental engineers and data scientists
The present work was developed during the second semester of the 2020/2021 school year, and was part of the MoST project, Modeling, Querying and Interactive Visualization of Spatiotemporal Data[1] (PTDC/CCI-INF/32636/2017).

FIRELab
- Wildfire Imagery Analysis and Processing

António Fernandes, 92880
antoniojorgefernandes@ua.pt
David Morais, 93147
davidmorais35@ua.pt
Francisca Barros, 93102
francisca.mbarros@ua.pt
Luís Silva, 88888
luisfgbs@ua.pt
Mariana Ladeiro, 92964
marianaladeiro@ua.pt

[1] https://www.inesctec.pt/uploads/inline/Ficha%20de%20Projeto_Most.pdf

Abstract

The following report portrays the final project carried out for the course of Informatics Project (PI, standing for Projeto em Informática) of the Informatics Engineering degree of University of Aveiro.

FIRELab is a project that fits in the MoST project, financed by the Foundation of Science and Technology, in which the Department of Electronics, Telecommunications and Informatics (DETI) and the Department of Environment and Planning (DAO) of University of Aveiro as well as the Institute of Systems and Computer Engineering, Technology and Science (INESC TEC) of Porto participate in.

The FIRELab project was created in order to assist the MoST project by improving old and creating new essential tools to process fire related images and videos obtained using UAVs. These images were obtained during experimental fires, conducted in a controlled environment in order to get not only the video data but also air quality data.

As a consequence the main objectives of the FIRELab project were to develop a tool to automatically classify the vegetation, improve existing tools to segment the burnt area, develop a module to georeference polygons obtained from the segmentation process and overlap them on maps, export data using specific formats to be used on other tools and connect all these tools under a single interface.

The present report covers the full development of this project, describing the current work available on the scope of the project, its theoretical backbone, the steps carried out in order to meet the objectives and the final results.

Keywords:
Segmentation, Polygon Extraction, Georeferencing, Vegetation Classification, Spatio Temporal data, Orthophoto, Fireprogression

Resumo

O presente relatório retrata o projeto final realizado para a unidade curricular de Projeto de Informática (PI, sigla de Projeto em Informática) do curso de Engenharia Informática da Universidade de Aveiro.

O FIRELab é um projecto que se enquadra no projecto MoST, financiado pela Fundação da Ciência e Tecnologia, em que o Departamento de Electrónica, Telecomunicações e Informática (DETI) e o Departamento de Ambiente e Planeamento (DAO) da Universidade de Aveiro participam, bem como o Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC) do Porto.

O projeto FIRELab foi criado para auxiliar o projeto MoST, melhorando as ferramentas antigas e criando novas ferramentas essenciais para processar imagens e vídeos relacionados a incêndios, obtidas através de UAVs. Essas imagens foram obtidas durante incêndios experimentais, realizados em ambiente controlado, a fim de obter não só os dados de vídeo, mas também os dados de qualidade do ar.

Como consequência, os principais objetivos do projeto FIRELab eram desenvolver uma ferramenta para a classificação automática da vegetação, melhorar ferramentas existentes para a segmentação da área queimada, desenvolver um módulo de georreferenciação de polígonos obtidos através do processo de segmentação e sobrepô-los em mapas, exportar dados em formatos específicos para serem usados noutros sistemas e conectar todas estas ferramentas numa única interface.

O relatório presente aborda todo o processo de desenvolvimento do projeto, descreve o trabalho atualmente disponível no contexto do projeto, a análise teórica, os passos executados de forma a atingir os objetivos e os resultados finais.

Palavras-chave:
Segmentação, Extração de Polígonos, Georreferenciamento, Classificação da Vegetação, Dados Espaciais Temporais, Ortofotomapa, Progressão do Fogo

Acknowledgements

The group cannot express enough gratitude to our supervisores Professor José Moreira and Professor Hélder Zagalo as well as our co-advisor Patrícia Fernandes and Bruno Silva for their continued support, encouragement and trust.

António Fernandes
David Morais
Francisca Barros
Luís Silva
Mariana Ladeiro

Table of Contents

List of Figures And Tables

Abbreviations

VM Virtual Machine
HTML HyperText Markup Language
CSS Cascading Style Sheets
JS Javascript
RDBMS Relational Database Management System
ORM Object Relational Mapping
GeoJSON Format for encoding a variety of geographic data structures
UI User Interface

Chapter 1

Introduction

1.1. Context

This project fits in the MoST Project for which various field experiments were conducted in order to capture high resolution RGB and thermographic images from controlled fires, using drones.

The work focuses on implementing a set of tools to transform images and videos (raw data) into reliable georeferenced data, ready to be used by investigators, environmental engineers and data scientists. The videos and images of the site are taken by drones. In order to create the orthophoto, a large number of overlapping pictures of the site prior to the fire, taken orthogonally to the ground, are needed. Through multiple passes of a drone over the area these images can be captured and then put through third-party software that compiles the results and produces the orthophoto. The videos captured throughout this project have been of planned controlled burns. On site there is a team of firefighters performing the controlled burn and a vehicle-mounted weather station monitoring the weather conditions and other metrics like air quality.

It is not within the scope of our project to assist in the creation of the initial raw material used for our processes, the images and videos of the site and fire and the creation of the orthophoto are of the responsibility of the user.

The main challenge consists in modelling and processing spatio-temporal data, meaning, modelling not only the burnt area or the front-line that defines the front of the fire in a specific instant, but also it's evolution over time. The video data being used was obtained using UAVs during controlled fire experiments.

1.2. Motivation

The problems to confront with the project fall in the hard work carried out by specialists, in which the tools offered are not keeping up with the evolution of technology, meaning, because of their outdated assets, specialists' work is hindered and takes too much time. Diving deeper into these problems, one is the characterization of vegetation, which is made from aerial images captured before the fire, in the interest

areas, that is currently done manually, making the whole process quite slow. Concerning the process of segmenting the burnt area through the videos obtained during the experiments, the irregular behaviour of flames, as well as the obstruction of crucial parts of information in one or more frames of said videos, because of the smoke, embodies another problem. Moreover, the majority of available tools up to today for the type of work described are command-line or make use of an unresponsive interface.

With this in mind, the motivation of the work developed centers its attention on improving the already implemented tools as well as bringing new ones which will consequently ease the effort made by these specialists by streamlining and automating the processes.

1.3. Goals

The goal of the project was to develop a set of tools as well as implement and adapt already developed tools and aggregate them in a ready to use web application of processing support, analysis and visualization of data on the forest fire propagation and gas emission to the atmosphere.

The main tasks to accomplish were to develop an application to characterize the interest area that allows transforming the photos taken before and after the fire into orthophoto maps, dividing the interest area in cells in order to characterize the vegetation in each cell before the fire, using automated methods to minimize the user's input; adapt a already developed method to extract the geometry of the burnt area or the fire front line in RGB images and realize sequences of images extracted from videos; implement methods to allow transformation of local coordinates in georeferenced data and export data to modelling tools of fire propagation and smoke dispersion; implement a tool for data analysis that allows to visualize and compare the original images with the processed data and export animations in video format; and lastly, define the project architecture and implement a demonstration application that integrates and aggregates all of the tools referred above in a easy to use application, while constructing an appealing user interface. All of these tools are considered modules, deeply decoupled that can be used by themselves, without any links to the other existing tools. The application developed is the only part of the project that aggregates the modules but maintains their decoupled nature.

1.4. Document Structure

Besides the introduction, this report has - more chapters. In Chapter 2 the state of the art as well as related works is described and how all of the presented systems correlate to the project developed. Chapter 3 details the requirements elicitation, requirements and the system architecture. Chapter 4 describes the theoretical analysis of each module developed. In Chapter 5 a more detailed explanation of how the system was implemented as well as decisions made, are described. The tests conducted as well as

the results obtained are presented in Chapter 6. At last, Chapter 7 concludes the report making an overview of the work done and future work.

Chapter 2

State of the Art

The aim of this section is to give a more detailed description on the scope in which the project fits in. Thus, it will enumerate the various projects and systems that intersect with the goals of the project to be carried out as well as the technologies used in the implementation.

2.1. Related Work

This section presents the work conducted until now on the domains that intersect with this project. There are few tools already implemented, which our project used as a base to start developing an improved version. An overview of those systems is presented below.

2.1.1. MoST Project

The MoST project consists of a set of features that allow preprocessing and extractions developed by IEETA, INESC TEC and GEMAC. The general purpose is to analyze tools to study the behaviour of spatiotemporal phenomena in various domains from biology to earth sciences. Developing "methods and tools to transform raw data from sequences of images and videos into meaningful data that can be modelled and represented in a database or information system, as well as efficient methods for querying, visualization and analysis of data about the location, movement and morphology of spatiotemporal phenomena"(MoST, n.d.) are included.

2.1.2. FARSITE

FARSITE (Finney, 1998) is a fire growth simulation modeling system that can compute wildfire growth and behaviour under heterogeneous conditions of terrain, weather and fuels. It incorporates existing models for surface fire, crown fire, spotting, post-frontal combustion and fire acceleration into a two-dimensional fire growth model. This tool is widely used by the United States Forest Service and other federal and state land management agencies. According to their website, FARSITE

should only be used by "users with the proper fire behavior training and experience" since it is a very complex platform. FARSITE produces as output characteristics of the fire behaviour, for example spread rate, as well as environmental conditions like solar irradiance.

2.1.3. DISPERFIRE

DISPERFIRE (Valente et al., 2007) is a smoke dispersion modelling system used to estimate the 3D concentrations of pollutants emitted during a forest fire event. DISPERFIRE is based in the diagnostic wind model called NUATMOS as well as a Lagrangean dispersion model. This system is usually applied when there is a need to obtain results quickly and its model is more adequate for small scale burns.

2.1.4. ArcGIS

ArcGIS is a geographic information system (GIS) that works with maps and geographic information. It allows the creation and use of maps, analyzing mapped information, using maps and geographic information in other applications and computing geographic data. Although this is a system with much use by the target users of our project, it was not needed to develop the functionalities effectively.

2.2. Technology

This section describes the technologies considered and the ones used to implement the solution in the project, as well as the reason why these technologies were chosen.

2.2.1. OpenCV

As a starting point, it was made available an extraction tool that extracts video frames and segments the burnt area as well as conversion of coordinates in the image (pixels) to global coordinates (lat and long), written in C++ and using OpenCV. OpenCV is a multi platform library that contains image processing models.

In the module regarding the fire segmentation, the Watershed algorithm of the OpenCV library was used on a certain image to be segmented. This algorithm is extremely useful for image segmentation and can be viewed as a topography surface selection algorithm where, in the grayscale of the provided image, pixels denoted with high values are considered peaks and pixels with low values are considered valleys. According to the OpenCV documentation, the philosophy behind the watershed algorithm is as follows:

> “You start filling every isolated valleys (local minima) with different colored water (labels). As the water rises, depending on the peaks (gradients) nearby, water from different valleys, obviously with different colors will start to merge. To avoid that, you build barriers in the locations where water merges. You continue the work of filling water and building barriers until all the peaks

are under water. Then the barriers you created gives you the segmentation result." (OpenCV, 2021)

However, this approach gives an over segmented result due to irregularities in the image, so OpenCV improved this algorithm by needing to specify which are the valley points to be merged and which are not. The foreground will be represented with one color, or intensity, and the background with another color, making the segmentation a lot more accurate, defining the areas that are burning in contrast to the rest.

The OpenCV library was also used when dealing with the fire progression module. Since there was a need to visualize, in a map, in the location on which the data was obtained, how the fire progressed, homography was utilized. Homography is a transformation that allows mapping of the points in one point to the corresponding point in another image. With homography from OpenCV it is possible to convert any points in a given image to the corresponding coordinates (latitude, longitude).

2.2.2. GDAL

Given the system deals with large sized files, there was a need to find a technology that would allow these files to be reduced in size in order to increase the system's performance. Additionally, there was also the need of a tool that not only read the metadata of the orthophoto geotiff file (regarding the vegetation module) but also allowed to work on/with the latter with dexterity. GDAL was recommended, thus used to deal with both the orthophotos and geospatial data analysis and storage. Geospatial Data Abstraction Library, GDAL, is a translator library for raster and vector geospatial data formats.

Specifically, this library was used in the classification of the vegetation module, when dealing with importing an orthophoto, to reduce both quality and size of the file. Furthermore, the gdal_translate method, which converts raster data between different formats, was also made use of in the same module. "The gdal_translate utility can be used to convert raster data between different formats, potentially performing some operations like subsettings, resampling, and rescaling pixels in the process." (GDAL, n.d.)

2.2.3. Django

Regarding the decision on the technology to implement the system itself we first needed to make a decision between developing a web app or a desktop app. Both alternatives had their own advantages as well as disadvantages.

The main downside of implementing a desktop app was the need to deal with versions and operating systems as well as adding a step to the user (that would need to install the application). Besides the fact that the group didn't have a broad knowledge regarding the creation of a desktop app, there was also the problem of time management and whether the available time would be enough.

The main apprehension of developing a web app fell on how the server would behave with the functionalities we would have to implement, especially because of the fact that they would have some sort of dependency to other systems. Additionally, there was also the problem of how much time it would take for the server to import the files needed to interact with the platform. However, after making some tests we found that there would be no problem with the server interacting with other systems. Moreover, it was decided that the group would develop a web app, since performance was not an objective of the project and it didn't constitute an impediment for the development.

Having the web app decision in mind, the group chose to use a framework to streamline the developing process. Between the options considered, ReactJS, AngularJS and Django were the focus ones. Since Django uses Python, a programming language that the group felt comfortable in, it was determined that the latter would be used to implement both back and frontend.

2.2.4 Leaflet

As a starting point it was proposed to explore Leaflet and OpenLayers for the visualization of data on maps for the fire progression module.

OpenLayers is an open-source JavaScript library that displays map data in web browsers. It reads coordinates in the Longitude, Latitude format which can be a downfall since most map libraries use the inverse, meaning, Latitude, Longitude.

Leaflet is a lightweight open-source JavaScript library that creates interactive maps. Leaflet allows the rendering of polygons, which was needed to model the fire progression. Because of this and due to the fact that Leaflet was an easier to use library the choice was made. Leaflet was then used to enable the animations of the polygons that indicate the progression of the firefront.

2.2.5 GeoDjango

In a first approach, it was thought that there would be a need for an intermediate layer to mediate communication between the database and the client applications, like Geoserver. Geoserver is an open source server used to publish data that are on the database as web services, allowing sharing, processing and editing geospatial data. However, since Django was chosen, GeoDjango was also an option.

> "GeoDjango is an included contrib module for Django that turns it into a world-class geographic Web framework. GeoDjango strives to make it as simple as possible to create geographic Web applications, like location-based services. Its features include: Django model fields for OGC geometries and raster data." (Django, n.d.)

Because of the fact that with GeoDjango we could still obtain the data the same way, it would eliminate one step in production, meaning the configuration of the middleware, Geoserver.

2.2.6 Docker and PostgreSQL

Regarding the deployment tools, Docker was utilized creating a Dockerfile for the webapp in conjunction with docker compose to run the application as well as the PostgreSQL database dependency with the PostGIS extension.

In terms of data storage, PostgreSQL was chosen as the relational database management system (RDBMS) to save all information linked to the service developed - from users to projects and their unique files.

2.3. Summary

The existing segmentation tool was a starting point to the development of the segmentation tool implemented in our project. Furthermore, this system is not supposed to innovate in any particular field but provide a set of tools that can streamline the pipeline of ambiental engineering as well as providing a much needed connection between the outputs and inputs of the remaining systems.

Finally, it's important to mention that a very in-depth research in these models wasn't conducted, since this doesn't fit in the objectives of the project and are specialized ideas that weren't needed in order to meet the goals. However, there is a general idea of every single existing platform already developed and referred above since the project makes use of them.

Chapter 3

System Requirements and Architecture

This section presents the system requirements that describe the prerequisites needed in order to know what to develop and the architecture.

3.1. System Requirements

This section describes the system requirements specified by the stakeholder and the goals of the project. The following subsections present the requirements elicitation, context description, actors of the system, use case diagram and description and finally non-functional requirements.

3.1.1. Requirements Elicitation

In order to gather information regarding the system requirements, not only was there extensive research and study done on other similar technologies and systems already developed, but also several informal interviews conducted with an environmental engineer. The first helped to understand what the developed system needed in order to function and to be a better and more valuable piece of work, allowing it to stand out when being compared to similar ones. The ongoing interviews helped with shedding light on what the system should and shouldn't do as well as setting boundaries and priorities on the features to be developed.
The following requirements were requested by the stakeholder:

- The user should be able to create a grid, in the orthophoto imported;
- The application should allow the user to choose the cell size when constructing a grid;
- The application should be able to characterize the vegetation in a certain grid, in the most automatized way;
- Once the characterization of the vegetation is concluded, the application should return a file in the ASCII format (.ASC). This format is required in order to use this file as input in a tool that is not part of FIRELab;

- The user should be able to select the burnt area and the area to disconsider in a given image or frame provided by the user, so that the polygon that represents the burnt area can be extracted;
- The user should be able to visualize the fire progression;
- The user should be able to export the obtained data to the format of fuel map (resulting grid needed) and DISPERFIRE (video and grid needed).

With all of this in mind, it was decided that each main goal would be divided into modules. These modules are: vegetation classification, fire segmentation, fire progression. As mentioned before, these modules are decoupled and it's implementation should keep this nature, as much as possible.

3.1.2. Context Description

In this section, a brief description on how the system is expected to perform for each actor is discussed.

The researcher - represents the researcher/ environmental engineer/ data scientist that will use the application - should be able to register in the platform first, in order to have access to the functionalities available. This researcher can create new projects, as well as delete an existing project. After creating a project, a researcher can segment images and extract the resulting shape, classify the vegetation, turn points of an image into georeferenced points, visualize and compare models and export data.

The system manager will take care of maintaining the system and manage database quotas of the registered accounts.

3.1.3. Actors

The use of the system is targeted to a very specific group of users, whose knowledge on environmental fires are medium to advanced and have a background of making use of similar functionalities. However, the interface was designed in a way to make the application perceptible and enjoyable to any type of person, but it should be clear that some terms and features on the system are not very likely to be recognized by everyone.

The actors are described in the following list:

- Researcher: Represents the researcher/environmental engineer/data scientist, having access to all the functionalities available on the system.
- System manager: Deals with the system maintenance as well as managing user's accounts.

3.1.4. Use Cases

Figure 1 presents the overall case model of the system, representing the usage of the same. Each module is divided, showing a more detailed description of the functionalities available.

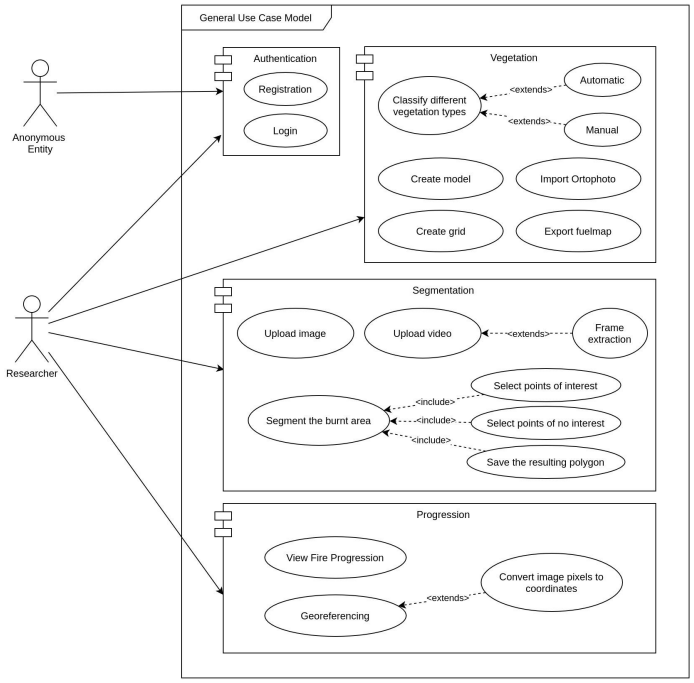

Figure 1: General Use Case Model

- Registration: allows an unknown entity to register itself in the system, giving a few personal information.
- Login: the registered researcher is able to enter the system.

Vegetation Module:

- Create model: allows the researcher to create a model in which the classification will be based upon.
- Import orthophoto: the researcher can import a certain orthophoto to start working.
- Create grid: allows the researcher to create a grid by selecting two points in the orthophoto.
- Classification of different vegetation types: allows the researcher to, given an orthophoto and grid, characterize the different vegetation types in said grid, either manually or automatically.
- Export fuelmap: the researcher can export the resulting vegetation classification.

Segmentation Module:

- Upload image: the researcher is able to upload an image to start working.
- Upload video: the researcher can upload a video and select the number of frames per minute to be extracted as well as the starting real time for that extraction to begin.
- Segment the burnt area: given a video or image, the researcher is able to segment the burnt area, by selecting points of interest and no interest as well as manually correcting some. Lastly, the researcher is allowed to save the resulting segmented polygon.

Progression Module:

- Convert pixels to coordinates: allows the researcher to georeference the polygon generated in the segmentation module.
- View fire progression: with the segmented images, the researcher can view how the fire progressed overtime in a map, in animation mode.

3.1.5. Non-Functional Requirements

The non-functional requirements are presented in the list below:

- Reliability: Since dealing with real and important data, and having functionalities that require the system to be up for a period of time without failing, it's important that error-handling is approached with caution. In case of error, the system should be responsive in order to not leave the user in the dark;
- Efficiency: The system will be a web application, thus it should provide the intended results in the most efficient way possible, by not consuming unnecessary resources in the user's computer. For this, good practices of web development microservices should be followed.
- Security: The system uses authentication and authorization in order to make sure not everyone can access the given functionalities;
- Interoperability: Since the system will need to integrate other services and applications to perform the intended functions, every operation that depend on those should be performed in the most transparent way, without being a barrier for the user;
- Usability: Although the system will be used by a very specific group of users it is still quite important and a goal that an easy and intuitive system is developed - since this was a characteristic lacking in other platforms;
- Compatibility: The system should be compatible with the largest amount of operating systems and versions possible.

3.2. System Architecture

The subsections that follow provide an overview of the system's architecture - its domain models, both physical and technological, as well as the interactions between its components.

3.2.1. Physical Model

The physical model, presented in the image below, shows a high-level view of the system's physical architecture, its components and their interactions and implementation.

On the user side, through a web browser, the user will be able to access the platform where the server is hosted, by using HTTP protocols.

The server that will host the platform will be a VM on Microsoft Azure, containing the web server and database associated with it, as well as the modules that allow for the processing and analysis of imagery.

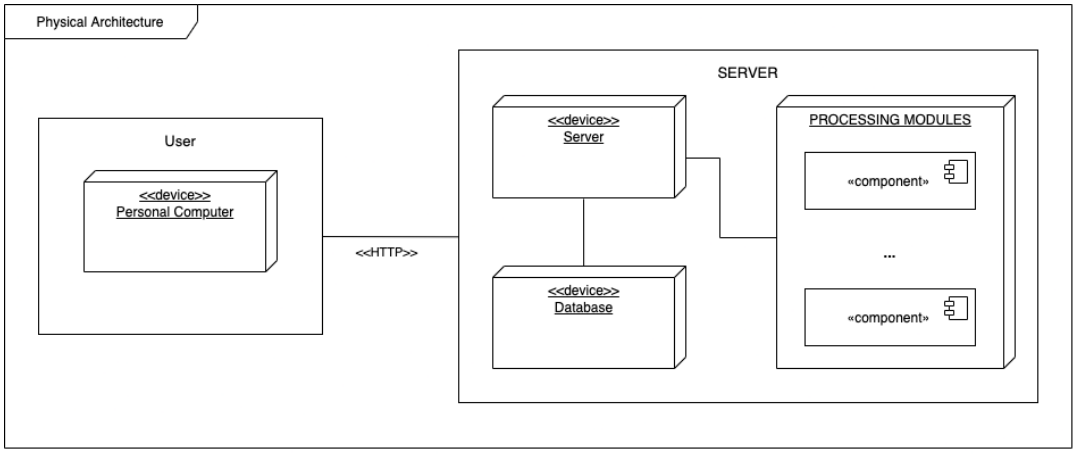

Figure 2: Physical Architecture

Every other instance should not be able to access the relational database. It just has to be exposed to the Django Server, who will manipulate it accordingly.

3.2.2. Technological Model

The technological model provides a view over the technologies used by the system. The figure below illustrates them, and shows how they interact.

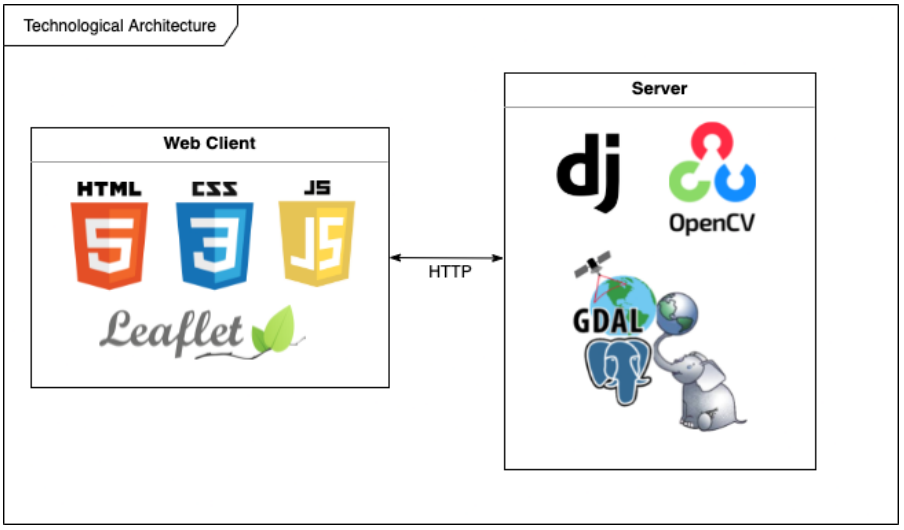

Figure 3: Technological Model

Django was chosen as the framework to allow the construction of an intuitive and responsive web platform, both on the frontend and the backend, because it integrates the Python language with HTML, JavaScript, and CSS. Data storage is handled on

the server side via the Django ORM, who manipulates the dedicated PostgreSQL instance.

All of the other tools and libraries depicted in the diagram have previously been explored in earlier chapters. In the following chapters, the way they interact with one another, as well as the reasons for their choices in the context of the project, will be discussed.

Chapter 4

Theoretical Analysis

This section provides a deeper insight into the theoretical analysis of the scope of the project including data collection and database. The next subsections are divided according to the overview of the modules of the project: vegetation classification, segmentation and georeferencing (included in fire progression).

4.1. Database

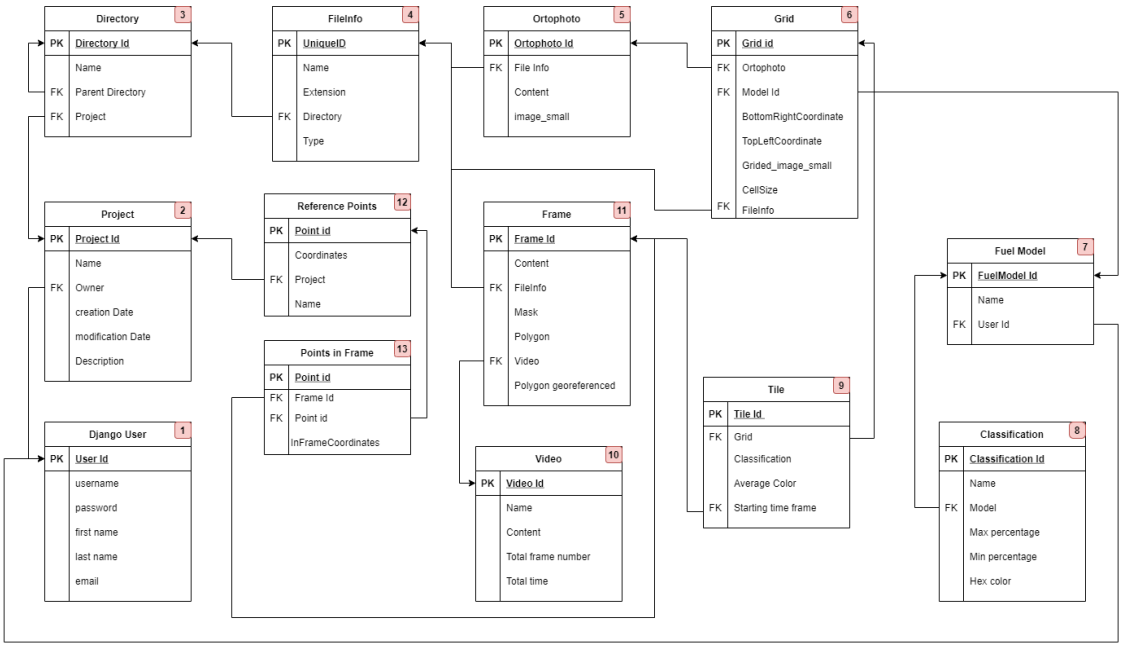

Figure 4: Database model

First we introduce, with tables 1 and 2, the notion of Users followed by the notion of Project. The choice of creating a clear rigid structure within each project with distinct, immutable folders that create separation between different user handled components of a project such as images, orthophotos and grids, is implemented in the structure through

the creation of tables 3 and 4. This allows for quick and easy implementation of Directories in the UI, increasing it's organisation and overall user experience.

Table 5 holds the orthophotos imported by the platform's users and has a One-To-Many relation with table 6, allowing for multiple grids to be defined over a single orthophoto. In turn, the grids (table 6) also have a One-To-Many relationship with table 9, that holds the Tiles, that represent each of the cells of the grid, and a Many-To-One relationship with table 7, that holds all Fuel Models created by a user.

A fuel model is composed of one or more individual classifications (table 8) with an associated name, percentage range and color, and so another One-To-Many relationship is defined between table 7 and 8.

Videos imported to the platform will be stored in table 10 and the frames resulting from the frame extraction process will be stored in table 11, which establishes a One-To-Many relationship between the two tables.

Reference points, stored in table 12, are used in the georeferencing process and are marked over the frames extracted from a video. This table holds information regarding the name given to the reference point and it's geographical coordinates while, table 13, holds information of the frames in which each point is marked as well as it's position within the frame, effectively establishing a relationship between the reference points, table 12, and the frames, table 11.

4.2. Vegetation Classification

Third-party fire progression modelling tools such as FARSITE model base their processes on the initial conditions of the research region. The data required includes the area's vegetation density which can be provided to this software via a prepared file.

```
84
142
615671
4582474
5
51.0 51.0 51.0 21.0 51.0 21.0 51.0 21.0 1.0 99.0 1
21.0 51.0 51.0 51.0 51.0 21.0 51.0 21.0 51.0 99.0
99.0 21.0 1.0 99.0 99.0 99.0 99.0 1.0 51.0 51.0 51
51.0 51.0 51.0 51.0 51.0 21.0 51.0 51.0 21.0 99.0
51.0 51.0 99.0 21.0 99.0 51.0 21.0 21.0 51.0 99.0
99.0 99.0 99.0 99.0 99.0 99.0 1.0 21.0 51.0 99.0 2
1.0 99.0 21.0 99.0 51.0 51.0 51.0 51.0 51.0 99.0 9
99.0 99.0 51.0 21.0 51.0 51.0 21.0 51.0 21.0 99.0
51.0 99.0 99.0 99.0 99.0 99.0 99.0 99.0 99.0 51.0
99.0 51.0 99.0 51.0 51.0 21.0 51.0 1.0 51.0 1.0 99
51.0 21.0 99.0 1.0 99.0 51.0 21.0 99.0 99.0 99.0 9
99.0 99.0 99.0 99.0 99.0 99.0 99.0 99.0 51.0 51.0
```

Figure 5: Rigid file formatting of FARSITE inputs

In figure 5 it is displayed the rigid format that the file must follow, in the header the following information must be present: number of rows, number of columns, grid's origin x coordinate (UTM), grid's origin y coordinate (UTM), and cell size (meters). The content columns*rows matrix in which each position (i,j), $i \in [0, \text{columns}]$, $j \in [0, \text{rows}]$ contains the value of the cell's attributed classification.

Up to this moment this process has been done manually, however the intent is to provide the user with a tool to assist in the classification of the area of study while still allowing for manual interaction with the results for tweaking/adjusting in an effort to reduce the time between data collection and fire modelling as much as possible.

4.3. Segmentation

To address the third-party software validation issue we must be able to, at any given frame, identify the firefront using images of the fire provided by the user. The polygons extracted from this analysis will be used as a baseline against which to evaluate the results produced by the fire evolution modelling tools. Similar findings validate that the external tool properly portrays and predicts fire evolution, whereas distant findings indicate that there may be flaws in the method.

4.4. Georeferencing

The segmentation process produces a polygon made up of the n+1 (because the first and last point must be the same) points that make up the geometry of the fire outermost bounds within a given frame. This, however, still does not allow for correct representation of the shape on a map as the current representation of the polygon is based on the frame's pixels and does not correlate to its geographical coordinates.

The process envisioned to establish the conversion from non-georeferenced to georeferenced polygons uses at least 5 reference points with known geographical coordinates to then extrapolate the geographical coordinates of the polygons extracted from each frame.

Chapter 5

Procedure

The objective of this chapter is to give a more detailed explanation on the procedure conducted in order to develop the system and meet the goals of the project.

5.1. Orthophoto Import

Regarding the upload of an orthophoto to be used as a base for the vegetation classification, there are two fundamental prerequisites which the user needs to satisfy - the overall file size must not exceed one gigabyte and the user account must have enough quota to allow the upload of said file.

Given these conditions are met, the orthophoto geotiff file can be sent to the server and saved in its file system. The first attempt of a functional upload feature was met with a enormous loading time, not only due to the file size - that despite being less than one gigabyte still took some time and memory to store -, but also that everytime the user needed to extract information or visualize said orthophoto the loading time required was considerably bigger than it was desirable.

To solve this problem, instead of saving the entire received file to the file system, we first needed to do some processing as follows: the four channels of the image are compressed using GDAL translate in order to reduce both the quality and size of the file while preserving the georeferenced pixels and file header which are needed to future analysis. In addition to that, a small jpeg image used as an alias is also created in the file system (also using the GDAL_translate method) with a 90% reduction in resolution of the previously computed geotiff. This considerably smaller image will be the one presented to the user as visual aid, so it was clear that quality was not an issue to be considered while compressing. It is also worth mentioning that this preprocessing process was approved by the project's stakeholder, once their usage of the orthophoto ranges the scale of a meter, hence not having the need for such high resolution geotiffs to be stored and worked upon.

Despite being a well thought solution, one problem still remains. This compression is carried out in the backend of the system, meaning that the whole file still needs to be sent, resulting in some unsatisfactory upload times. However we see that this

performance issue is outside the scope of the project and, despite being the drawback of using files with an excessive size, it only affects the upload times while not influencing the usage times of the developed system.

5.2. Grid

The notion of grid stems from the previously established need to classify the vegetation density. The grid will define the boundaries of the study area, define and hold the value for the size of each cell and allow for intuitive manipulation of the classification of each Tile that composes it.

5.2.1. Grid Creation

The system allows for the grid to be created over the imported orthophoto, relinquishing the need for a user created shapefile representing the grid and thus streamlining the process. After selecting the grid creating tool from the toolkit, any two clicks over the image will be seen as the definition of two opposite corners of the grid. With the value of the cell size desired by the user, we must then make verifications to assure that the length and width of the proposed grid are multiple of cell size and make the necessary changes to its dimensions if it is not.

Defining the grid will automatically start off the tile cutting process.

5.2.2. Cutting Tiles

Given a grid of dimensions, in meters, width *w* and height *h* and a cell size of *cell_size,* then the values for the number of rows *nrows* and for the number of columns *ncolumns* can be calculated as follows:

$$nrows = h // cell\_size$$
$$ncolumns = w // cell\_size$$

and every Tile within this grid is comprised of all pixels in a given position (c,r),

$$c \in [x_i, x_i + cell\_size], x_i = i * cell\_size \ \wedge \ i \in [0, ncolumns - 1] \text{and}$$
$$r \in [y_j, y_j + cell\_size], y_j = j * cell\_size \ \wedge \ j \in [0, nrows - 1].$$

And so, creating the tiles starts with leaps of size *cell_size* over the grid and the extraction of the *cell_size*cell_size* matrix of pixel values that will then be used to calculate the Tile's average color.

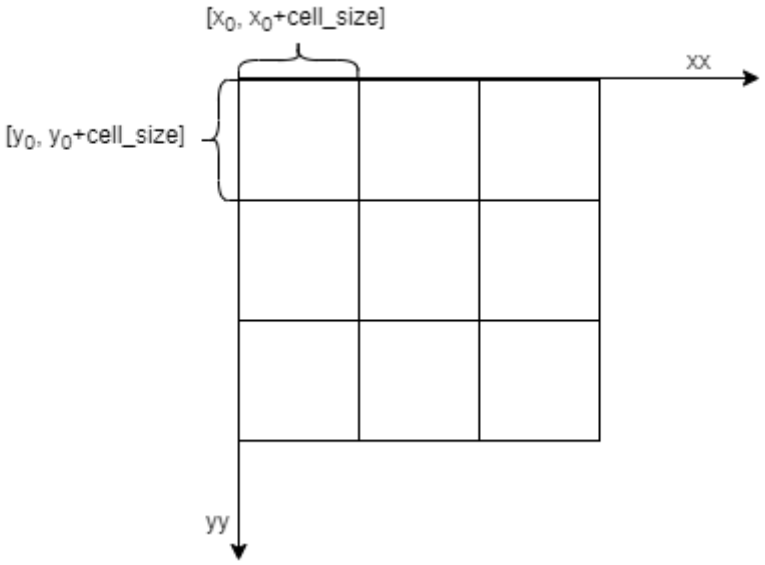

Figure 6: Visualization of tile extraction procedure

When storing the Tile, there is no need to store the matrix representative of the tile as the Tile's average color is a static value and would take up a lot of space essentially replicating the orthophoto.

5.3. Vegetation Classification

The standard Farsite model of classifications is as follows:

Class (Unofficial names)	Range (%)
No vegetation	0
Low vegetation	1-20
Medium vegetation	21-50
High vegetation	50-80
Deep vegetation	81-100

Table 1: FARSITE standard model of classification

In an effort to not lock the user into using this model or even Farsite itself, we have allowed for the creation of user made models in which users can discriminate the number of classifications as well as each classification's range and the hexadecimal color with which each of them will be represented on the grid. These models are user bound so once defined they can be accessed within any of the user's projects.

We support multiple grids within any orthophoto so that new evaluations and attributions can be made without interfering with previous ones and this frees up the user to create grids of different sizes over different areas which can each use different classification models. If the implementation did not allow for multiple grids to be associated with one orthophoto, the user would have to create multiple projects and

import the same orthophoto into each one of them in order to accomodate all of the desired grids even if they all share one single orthophoto. Thus, the current implementation promotes flexibility in defining the area of study and its classification while reducing stress on the server, reducing the number of imports needed and, consequently, the space and memory used in the server's filesystem.

5.3.1. Automated Classification

The process for the automated classification begins by calculating the average color for each of the defined classifications, which we will refer to as class_average_color (CAC) to better distinguish it from the Tile's average color which we will refer to as Tile_average_color (TAC).

With a generic grid, Grid, generic classification model, Model, associated with Grid and Class denominating a generic classification belonging to Model, to calculate Class' CAC, we select from the database all the tiles that have been classified as belonging to Class and attribute the average of these tiles' TAC to Class' CAC.

Then, after having the CAC for all classes that belong to Model, we iterate over all Grid's tiles and the algorithm attributes each one to the Class whose CAC is closest to this Tile's own TAC (nearest neighbour).

Running the automated classification process n times sequentially is possible and refines the results obtained. Manual user input between automated classifications has an even greater impact on the accuracy of results as moving tiles from one classification to another will affect both classification's CAC.

For the process to be possible, there must be at least one tile of the grid with a defined classification, which requires input from the user through manual interaction with the grid and its cells. It is not required for a classification to have attributed tiles but this does mean that that particular classification will have no presence on the visualization of the grid or fuel map. The image below exemplifies a manual classification that suffices to possibilitate the mentioned automated process.

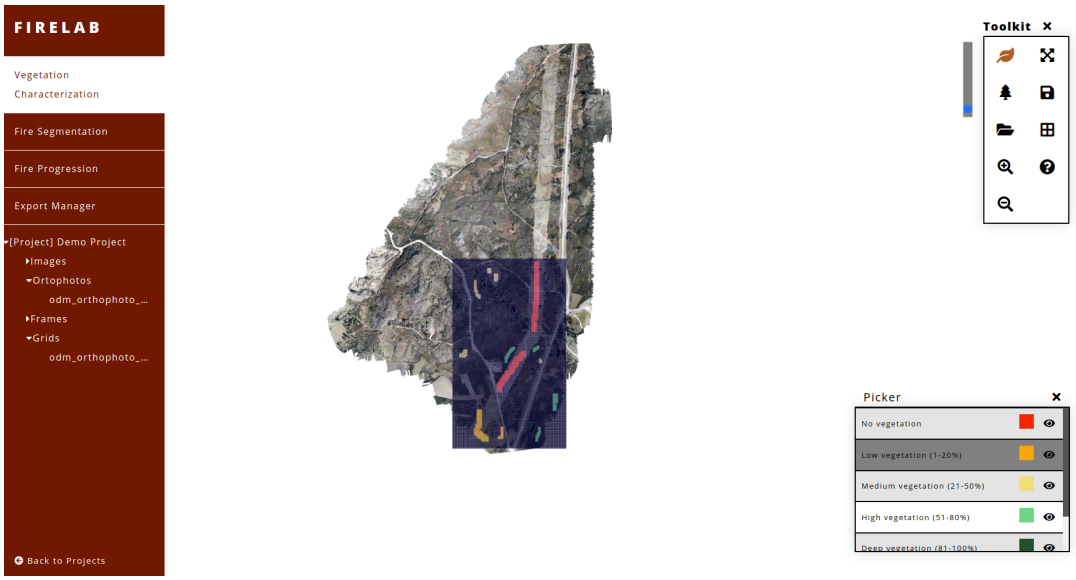

Figure 7: Example of manual vegetation classification

There are, however, shortcomings to the current implementation of this process. Dirt roads and asphalt roads are very distinct in color but both should be classified as having no vegetation. If an area of study comprehends both types of road simultaneously, because we calculate the average color of all tiles belonging to a classification, the algorithm will end up using a greyish color for the No vegetation class' average color and the process will yield very unsatisfactory results. Despite these drawbacks, we still obtain a very accurate classification of the area, as we can see in the following image, that continues the example of the above figure.

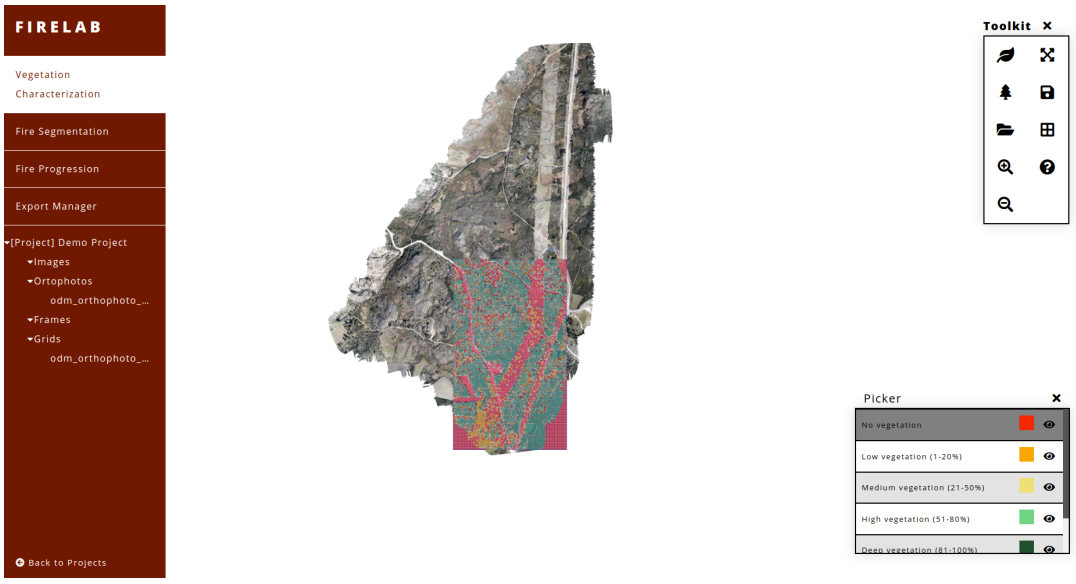

Figure 8: Example of automatic vegetation classification

5.3.2. Export Fuel Map

To create the file specified in section 4.2, it is required to have a grid already created and fully classified as the file can not have cells with no classification. Given that the previous criteria are met, and the grid defined is identical to the one specified in section 5.2.2, then the process for calculating the number of rows and number of columns is the one discussed in the aforementioned section and, using the information stored in the database upon the creation of the grid, BottomRightCoordinate($lat_{BottomRightCoordinate}$, $long_{BottomRightCoordinate}$) and TopLeftCoordinate($lat_{TopLefttCoordinate}$, $long_{TopLeftCoordinate}$), the desired origin's coordinates turns out to be

$$OriginCoordinate = (min(lat_{BottomRightCoordinate}, lat_{TopLeftCoordinate}), min(long_{BottomRightCoordinate}, long_{TopLeftCoordinate})) = (lat_{TopLeftCoordinate}, long_{BottomRightCoordinate})$$

This completes the information needed for the file's header, and then, for position (i,j), i ∈ [0, *ncolumns*], j ∈ [0, *nrows*], we can access the corresponding tile as its Id inside the database is composed of the tuple that represents its position inside the grid. With this, all there is left to do is access the tile's classification and add it to the file.

5.4. Segmentation

To be able to perform the segmentation process there is the need to previously import either an image or a video to work on. This process can be done in the toolkit, where after selecting the import icon, the user is provided with the choice of uploading either an image or a video. If an image is imported, then the process is linear, the image is saved in the filesystem of the server. However, if a video is imported then in the frontend the metadata of the video is analysed and the user is presented with the real-time at which the video began recording and the time at which it ends. Based on this, the user can then extract the frames by providing the real-time starting point at which the extraction should begin and the number of frames per minute to be extracted.

For example, in the following image taken from a real project on FIRELab, when selecting a video with length 5:27 minutes, we can see that the recording of the video was started at 9:49h of the 6th of March.

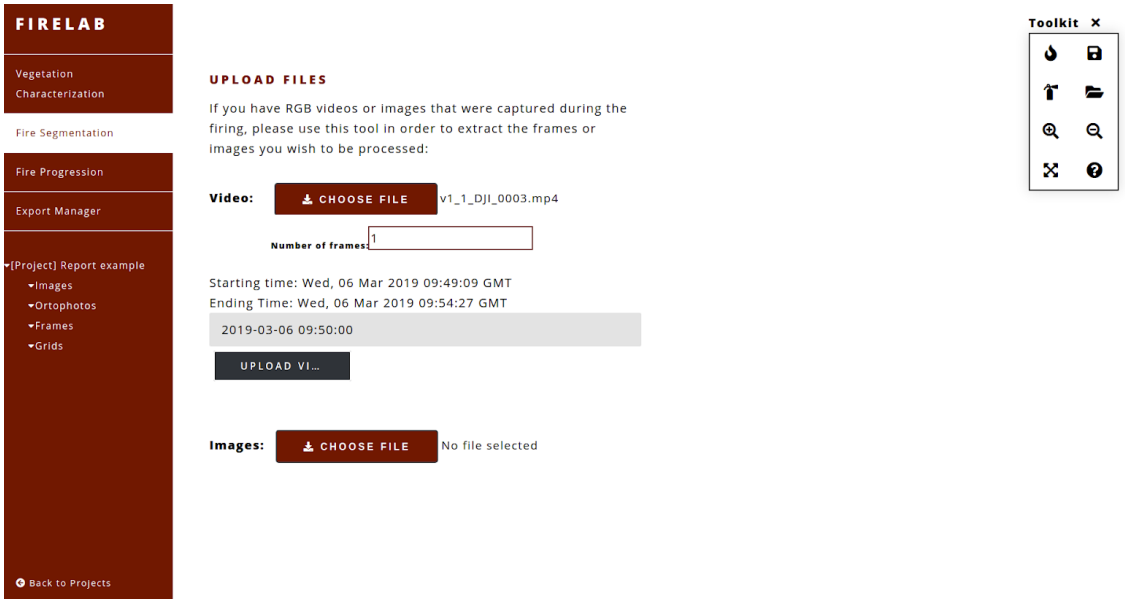

Figure 9: Upload video feature on the FIRELab website, used as an example.

In this case, if the user selects the starting time of the extraction at 9:50 of the same day, with a number of frames equal to one, then the system will ignore the first 51 seconds of the video to match the user input, and after that will save in the file system one frame every 60 seconds, resulting in the user being provided with four extracted frames - one for each minute starting at 9:50 and ending at 9:54. The last 4 seconds,

similarly to the first 51 seconds, are ignored so that a frame can be extracted every minute.

In order to segment a frame in the project, the icons in the toolkit must be used by the user to select either a point in the foreground (using the flame icon) or in the background (using the extinguisher icon). These points will be considered points of interest if they are marked as foreground, meaning that they should be marked in areas of the image where the fire is burning and, in opposition, the points marked as background will be considered points of no interest and such should be marked in areas where there is no fire.

Every time a click or drag is performed in the image, the server will receive the selected pixels, as well as the indication of whether they were marked as background or foreground. With this information it will then proceed to edit the mask inherent to every frame, altering the value of every clicked pixel to 255 if it is of interest or to 1 otherwise.

After this change is made, the Watershed (OpenCV, 2011) algorithm of the OpenCV library is called on the image to be segmented with the help of the mask. The purpose of the mask is to create borders in these floods by altering the values of certain regions using the points of interest and no interest. Changing the former to the highest value (255 in this case, so that these are considered as peaks) and the latter to the lowest value (in this case 1, and considering it a valley) creates these markers in the area, making it so that the segmentation is much more accurate, further defining the borders between areas that are burning and the rest.

The resulting image is then converted to base64 to be sent to the front end, so that feedback of the current state of segmentation is provided to the user, enabling a tweaking of borders.

When the user is satisfied with the current state of segmentation, clicking the save icon in the toolkit will save the resulting polygon to the database, triggering the process of generating the contours of the area. In a first approach, we tried to store the resulting polygon without pre-processing, however this frequently resulted in more than one polygon being generated due to small gaps in the segmented area, such as it can be seen below. Note that the segmentation made to exemplify the gaps on segmentation was done rather quickly as accuracy was not the objective.

Figure 10: Quick exemplification on gaps that can surge on the segmentation of an image

This issue was fixed by, before generating the contours of the final shape, applying a dilation followed by an erosion to the segmentation. What these morphological operations do is that in the dilatation, as the name suggests, the image is convolved by a kernel resulting in the borders of the area defined by the interest points being dilated around the area of no interest, expanding the shape and consequently closing the gaps that were causing the prior issues. The erosion that follows is analogous but with the opposite result, eroding the general shape according to the kernel, contracting the shape borders. This process is applied so that the outer border can be restored to match the ones in the image before the dilation, while still maintaining the previously mentioned gaps closed.

After this refinement is concluded, the vertices of the segmentation are computed using the function findContours provided by the opencv library and according to those we can create the Well-known text (WKT) (PostGIS, n.d.) representation of the geometry looping through the returned vertices. Furthermore, if more than a list of vertices is returned by the findContours function, then we solely consider the biggest found polygon in order to avoid a multipolygon WKT for being generated. It's worth mentioning that, after all the vertices are inserted in the WKTstring, we must append at the end the first vertice, due to the fact that wkt polygons need to have the same starting and ending point. This will generate a WKT string that represents the polygon of the segmentation in the following format:

$$POLYGON\ ((x_1\ y_1,\ x_2\ y_2,\ \dots\ ,\ x_n\ y_n,\ x_1\ y_1)),$$

$$n\ =\ size(shape)\ \wedge\ x\ \in [0,\ img_{width}]\ \wedge\ y\ \in [0,\ img_{height}].$$

With this list of points, we can then add the polygon to the frame entity in the database, using the geometric field Polygon provided by django and supported by the

postGIS database engine, so that a polygon representative of the segmented area can be stored alongside the frame in which it was segmented from.

Below is shown an example of how a concluded segmentation looks on the FIRELab website, and the generated polygon that it is stored in the database (it is worth mentioning that said polygon is stored vertically inverted due to the positive axis of the PostGIS system being the inverse of the one that opeCV library uses).

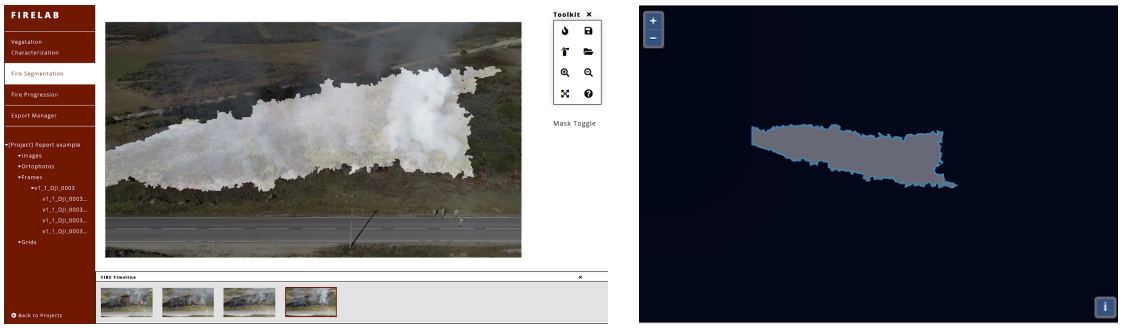

Figure 11: Segmented area and resulting polygon example

Despite our best efforts to improve the results using both dilation and erosion, hence facilitating the process for the user, the watershed algorithms perform much better in thermal images than in RGB ones. This is due to the fact that the value difference in border areas in the grayscale used by the algorithm are much more distinguishable and, consequently, the watershed is more accurate in delimiting those areas - even with less points both of interest and of no interest. This is further worsened by the fact that most of the RGB pictures are covered in grey smoke, making it so the values are even more similar. However we consider this as merely something the user must take into account and nothing that majorly debilitates the proper segmentation of an image, making it so that it takes some extra time for the user to get the same level of refinement.

5.5. Georeferencing

The segmentation process will produce a polygon with the shape of the burnt area. This shape will be measured according to the image's pixels and in order to display this polygon over a map, it's necessary to transform the pixel related points into geographic points. This process is called georeferencing and works by selecting points on an image (i.e. the width and height of a pixel on the image) and assigning them a latitude and longitude. With the pixel to geographic positions relation obtained with the opencv function findHomography(), it is then possible to create a transformation matrix that will convert any point of that image into the corresponding latitude and longitude. Applying this transformation to the polygon obtained from segmentation will make it possible to show it on a map and overlap it on an area of interest.

Figure 12: Inputting of coordinates(latitude, longitude) example

5.6. Polygon Visualization

The polygons produced from the segmentation of the fire-front images are indeed a fundamental piece of information, allowing the researcher to effectively understand how the fire evolved over time, and correlate this to simulations created using modelling third-party software.

As a result of developing a tool capable of generating animations containing the various polygons that map the fire evolution, the user will be able to easily evaluate and validate the data that FIRELab's software processed from raw, unprocessed imagery captured by the drones.

Being the module with the least amount of information accessible, or equivalent tools on which we could model our own design approach, it may be described as one of the most challenging to build. It represented a previously unavailable feature, employed unknown and untested frameworks and data-formats, and would significantly increase the productivity of the user.

After georeferencing the polygon, upon clicking on the play icon on the toolkit, the user will be presented with a map and two buttons, - one to start the animation and another to stop it. This last capability can be activated at any moment - that is, whether there are N or just 1 polygon displayed on the map.

The JavaScript library Leaflet was chosen to display the interactive map since it works well on most platforms and enables for expansions through its vast range of plugins. Moreover, JavaScript methods were designed and developed in order to display and layer the analyzed data. Georeferenced polygons may be thought of as simple geographical features, thus GeoJSON was adopted to conveniently depict them on the map.

When the user hits the appropriate button to see the animation, the underlying method will retrieve all georeferenced polygons on the current project, convert the

WKTs to GeoJSON format, and then add them as a Layer to the map if they are not already shown. As a technique to facilitate the animation process, there's always a little latency among each layer addition.

To summarize, if a user desires to examine the project's corresponding firefront progression (that is, see the animation), numerous calls are made, both to the backend (which accesses the database and obtains information) and to the frontend (that runs the appropriate script on the user-side, to process and display the data correctly).

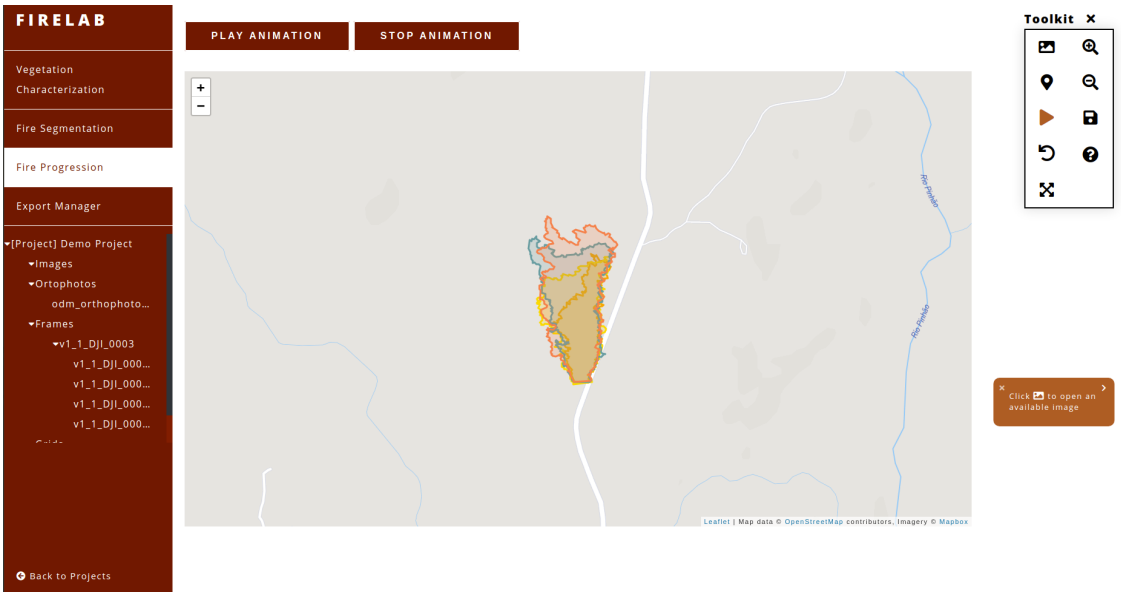

Figure 13: Visualization of the fire progression

5.7. Exporting to DISPERFIRE

The input file to the DISPERFIRE modelling tool is a formatted file with the header containing the highest and lowest numerical values, in seconds, for the cells time of first burn. The body of the file is a *columns*rows* matrix in which each position (*i*,*j*), i ∈ [0, *columns*], j ∈ [0, *rows*] contains the value of the cell's attributed start_time_frame (foreign key that leads to a frame).

To fill in this matrix with the appropriate values we must calculate the intersection between the georeferenced polygon and a new geometry that we will create representing each cell in the grid. To create the geometry we use the tile's position, the cell size and the grid's top left corner coordinates (in meters) to calculate each of the vertices. If the intersection comes back to false then the two geometries don't intersect and if comes back to true then they do and the reference to the frame containing the current polygon should be added to *start_time_frame*, if none have been added previously.

Basing our process on the georeferenced polygons extracted from the frames extracted from the videos we first use the outermost bounds (calculated from envelope) of the geometry referent to each frame, thus restricting the number of cells tested for intersection from the total number of cells in the grid, in the case of the 10*10 grid

represented below comes out to 100 tiles, to only the ones inside the outermost reaches of the polygon, 9 tiles, in an effort to reduce computational resources dispensed running this operation.

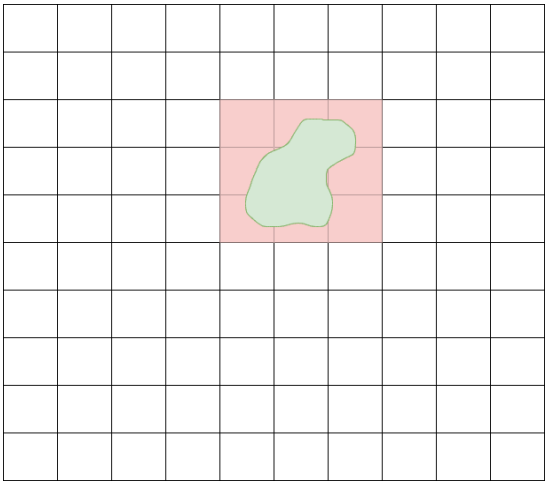

Figure 14: Tile Polygon Intersections in Computations for Cell's time of first burn

To further reduce the computational stress, we know that there is no need to test tiles that already have a start time frame value associated for intersections as what we are interested in is the time of the "first burn". So considering t=0 and t=15 and the two corresponding polygons, when calculating the intersections for the t=15 polygon, the area is reduced to the 4*6 grid of the outermost bounds and then from those, we never test for the tiles that already belong to *start_time_frame* t=0 further reducing the number of tested cells from $4 * 6 = 24$ to$24 - 9 = 15$.

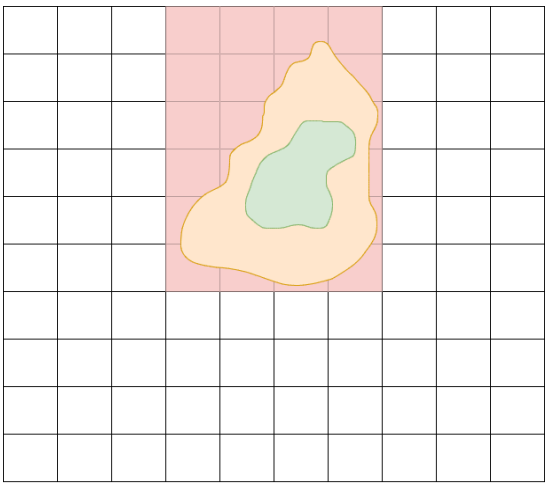

Figure 15: Tile Polygon Intersections in Computations for Cell's time of first burn (cont.)

For position (i,j), $i \in [0, ncolumns]$, $j \in [0, nrows]$, we can access the corresponding tile as its Id within the database is composed of the tuple that represents its position inside the grid. With this we can access the *start_time_frame* that should be in each position of the disperfire model that will be written and sent to the user.

The following figure is provided as a final example of the export to DISPERFIRE taking into account the orthophoto used for the vegetation classification example, as well as the video used for the segmentation and fire progression examples.

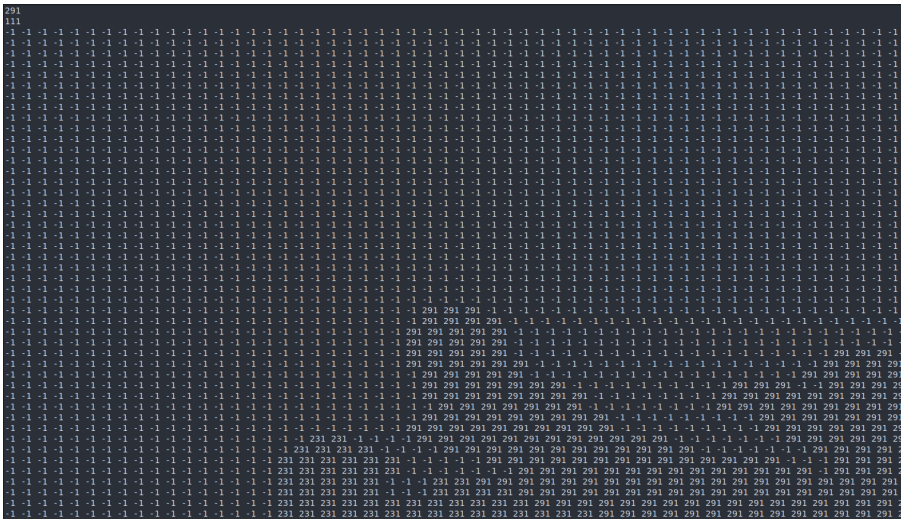

Figure 16: Extract from the generated file

Chapter 6

Tests, Overall Results and Discussion

6.1. User Tests and Results

Throughout all the stages of our project, the stakeholder's input was always present, in the Conception and Elaboration phases, defining the expected functionalities and, in the Development phase, the expected interactions and results with the implemented tools.

Having achieved the functionalities proposed, user tests were performed to validate the correct behaviour of the tools and algorithm implementations, file generation through the Export Manager and overall platform performance and usability.

The feedback from these tests exposed major usability issues with users reporting alarming experiences in regards to the lack of guidance when initiating the Vegetation Characterization and Segmentation processes as well as difficulty understanding tool usage. This was exacerbated by the lack of feedback when performing uploads or deleting items from the platform leading to undesired replication of the same process by the user. Finally, minor tweaks such as the unit of measurement and data format specifications (i.e. meters, latitude, longitude) were missing in many input fields and explanatory texts.

The issues reported in the tests were addressed and treated carefully in order to increase the user's satisfaction with the end result and meet the objective of creating an easy to use system. Thus, all the changes performed were validated by the stakeholder.

6.2. Overall Results

The research conducted, regarding the work currently implemented, indicated that the system being specified in this report was a much needed approach in order to update the tools available and improve the way they are presented to its users.

The efforts carried out throughout the development culminated in an end product that is capable of resolving the proposed challenges of automating the vegetation characterization, segmentation of the burnt area, georeferencing polygons obtained from the segmentation process, visualization of the fire progression, export data to fuel

map and DISPERFIRE format and the integration of all the aforementioned features in an intuitive UI with a high level of user satisfaction, thus achieving the final goal.

Chapter 7

Conclusions and Future Work

7.1. Conclusions

FIRELab is a software application that can aid in the processing of fire-related images and videos. It was aimed to support the MoST project by enhancing existing and developing new valuable tools that enhance both outcomes and the overall user experience and performance when conducting such activities.
Given that this project is perceived to be fulfilled, the group believes that the primary objectives established at the outset have been met with success.
Despite several obstacles along the development process, the team feels optimistic with the complete work completed and the expertise gained via the development of this project.

7.2. Future Work

In terms of future development, we consider that the overall product performance might benefit from enhancement in order to enrich the software developed.

Furthermore, as discussed in section 5.3.1 in regards to the Automated vegetation characterization algorithm, we are well aware of a crucial limitation in the implementation of our automated vegetation characterization algorithm, the inability to distinguish between two or more distinct colors belonging to the same classification.

By implementing the Classification Models we made an important step into a possible solution to this problem as, with this feature, a user could create a Classification Model tailored to the composition of a given area of interest in an orthophoto. For example, going back to dirt and asphalt roads, allowing the user to define a “Dirt Road” classification and an “Asphalt Road” classification that can be used to make the distinction between the two surfaces but correctly label both with 0% vegetation would be a simple fix to the situation. As of right now, the verifications for overlapping class percentages do not allow for the implementation of the previously mentioned solution but we predict that with some minor tweaking the implementation would be possible and, for this reason, we propose this implementation as future work.

References

Finney, Mark A. 1998. FARSITE: Fire Area Simulator-model development and evaluation . Res. Pap. RMRS-RP-4, Revised 2004, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p.

Valente J., Miranda A.I., Lopes A.G., Borrego C., Viegas D.X., Lopes M. (2007) Local-scale modelling system to simulate smoke dispersion. Int J Wildland Fire, 16 , (2), 196-203.

OpenCV. (2021) *Image Segmentation with Watershed Algorithm*. https://docs.opencv.org/4.5.2/d3/db4/tutorial_py_watershed.html

GDAL. (n.d.). *gdal_translate*. https://gdal.org/programs/gdal_translate.html

Django. (n.d.). *GeoDjango Tutorial.* https://docs.djangoproject.com/en/3.2/ref/contrib/gis/tutorial/

MoST. (n.d.). MoST | FCT Project. http://most.web.ua.pt/

PostGis. (n.d.) *OpenGis WKB and WKT.* https://postgis.net/docs/using_postgis_dbmanagement.html#OpenGISWKBWKT

PI 2021, *FIRELab.* Available online [22/06/2021]: https://github.com/itskikat/firelab

Leaflet JS *About Leaflet*. Available online [22/06/2021]: https://leafletjs.com/

Microsoft Azure *Cloud Computing Services.* Available online [16/06/2021]: https://azure.microsoft.com/en-us/

GeoJSON. Available online [22/06/2021]: https://geojson.org/