AIDERS

Deliverable 4.1 Multi-source data fushion algorithms

Coordinator Name: **Christos Panayiotou**

Coordinator Email: **christosp@ucy.ac.cy**

Project Name: **Real-time Artificial Intelligence for DEcision support via RPAS data analyticS**

Acronym: **AIDERS**

Grant Agreement: **873240**

Project Website: http://www.kios.ucy.ac.cy/aiders/

Version: **1.0**

Submission Date: **30/11/2020**

Dissemination Level: **Public**

The project has received funding from the European Union Civil Protection Call for proposals UCPM-2019-PP-AG for prevention and preparedness projects in the field of civil protection and marine pollution under grant agreement – 873240– AIDERS.

Funded by European Union Civil Protection

Contents

Executive Summary

The AIDERS project primarily aims at developing application-specific data analytics algorithms to harness the large volume of measurements that first responders are now able to collect through heterogeneous sensors (including visual, thermal and multispectral cameras, etc.). This deliverable details promising algorithms for data analysis (including feature extraction, classification, and prediction) that can be executed in real-time to enable relevant, reliable, timely, and simple information extraction and representation. The importance of data fusion algorithms, for information received by the heterogeneous input sources, is also emphasized especially for the modelling and prediction of the spread of the disasters and their impact.

1 Introduction

The increasing availability of ample sensory measurements, especially on board unmanned aerial vehicles (UAVs/drones) coupled with the increasing processing power of embedded and mobile devices has made advanced real-time decision-support systems feasible and very practical. In this deliverable we detail algorithms that process and extract important features that are directly or indirectly (through data fusion at a later stage) relevant to the information highlighted by first responders across Europe.

Hence we have split this deliverable to the following sections. Section 2 serves as a reference to the information required by first responders, while Section 3 provides an extensive review of the algorithms proposed to extract the relevant information and match the first responder needs. Finally, Section 4 concludes this deliverable and Section 5 provides a list of references.

2 Overview of requested information

For reference, we provide on Table 1 a summary of the required information as expressed by first responders of the consortium and those who completed our questioner. Extracting this information clearly required the processing of data from multiple sources and the fusion of the respective outputs to provide the necessary information.

Table 1 Short summary of the required information

1. Fire temperature prediction
2. Fire surface prediction
3. Flame height calculation
4. Smoke color detection
5. Vegetation type detection
6. Calculation of fire spread rate
7. Detection of dwellings
8. Detect and count people
9. Calculation of number of public buildings and parks
10. Calculation of number of roads
11. Prediction of fire evolution
12. Weather forecast
13. Electrical or explosion risk estimation
14. Route for rescue detection
15. Detect smoke

16. Thermal effects on nearby or adjacent buildings estimation and prediction
17. Hot spots detection
18. Smoke explosion risk estimation
19. Prediction of accidents due to falling objects, floor or roof collapse
20. Detect onlookers
21. Combustion toxicity calculation
22. Detection of changes in the toxicity and physiochemical structure
23. Extinguishing efficiency and foam carpet condition calculation
24. Detect nearby water sources
25. Fire propagation cloud mapping
26. Smoke concentration and opacity measurement
27. Onsite or nearby gases and liquids detection.
28. Predict toxic product leakages
29. Estimate explosion risk

3 AI algorithms for data processing

In this section we depict the 3 categories of algorithms for processing the available data and then elaborate on the algorithms within each category that will be used to extract the necessary information.

3.1. Image enhancement

The goal of image enhancement is to improve the usefulness of an image for a given task by improving the brightness the contrast and the appearance of the images. So, image enhancement techniques emphasize on accentuating certain image features, for subsequent analysis while reducing the enhancement of noise. There are different ways that these techniques can achieve the above goal such as enhancing the contrast among adjacent regions or features, removing noise, simplifying the image via selective smoothing or elimination of features etc. These techniques can be grouped into 3 different methods of enhancement: *Contrast manipulation*, *Spatial feature manipulation*, and *Multi-image manipulation*.

3.1.1. Contrast manipulation

The Contrast manipulation enhancement techniques contribute to the elimination of the inadequacies in the image contrast (too dark, too bright, too little difference between the brightness of features in the image etc.). To achieve the above goal, these techniques attempt to optimize the distribution of pixel values, over the radiometric range and so the DN expanding beyond their natural range to fill all the

values between 0 and 255 (8-bit image). This procedure can be done in a (i) linear way, (ii) by piecewise linear stretching or, (iii) by the histogram equalization (non-linear stretch). **Figure 1** visualizes the histogram expansion in each case. The algorithms for the above techniques are also attached.

i. Linear stretch
DN'=[(DN-Min)/(Max-Min)]*255

ii. Piecewise linear stretch
y=((L-1)/N)*C(x)

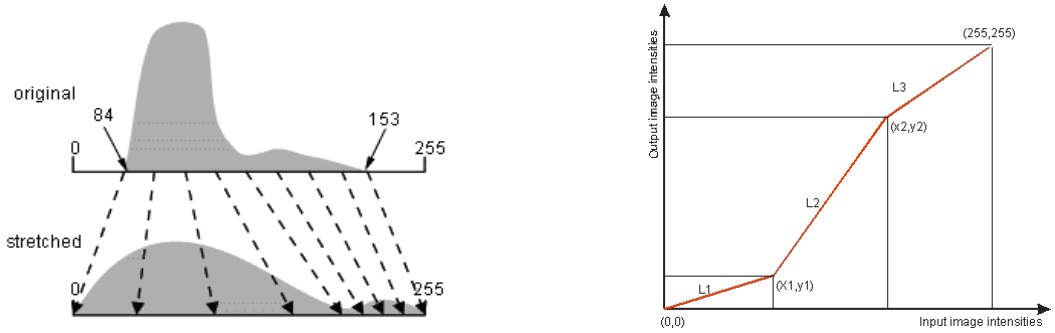


Figure 1: On the left image, is presented how the linear stretch method is working. In this case, min=84;max=153 while these values are becoming 0 and 255 respectively after the stretching. On the right, the piecewise stretching method is presenting where *L is the number of available brightness values; N is the total number of pixels and C is the cumulative histogram.*

Find below (***Figure 2***) an example where the linear stretch has been applied on a multispectral image. The result of this algorithm implementation on the multispectral image proved that these algorithms can be used to refine the given image, so that desired image features become much easier to perceive for the human visual system.

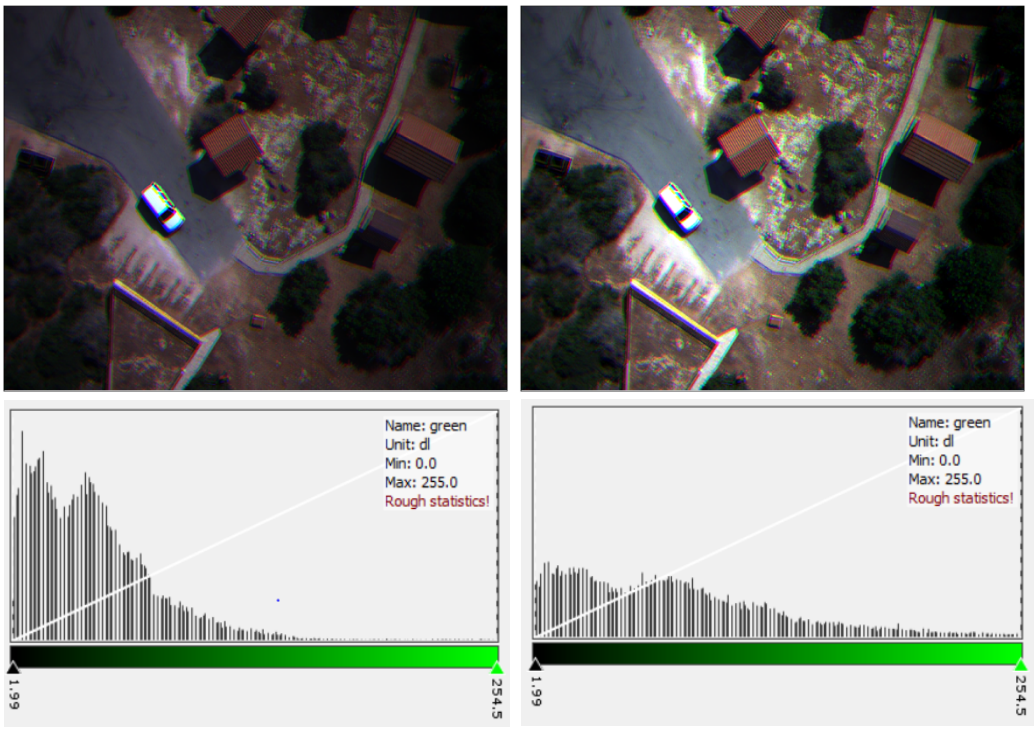


Figure 2: SNAP screenshot showing pre-and-post images with histogram stretching. The upper right image shows the linear stretched image. Displaying the upper left image would give a dark appearance without the desired contrast which has been achieved by the stretching.

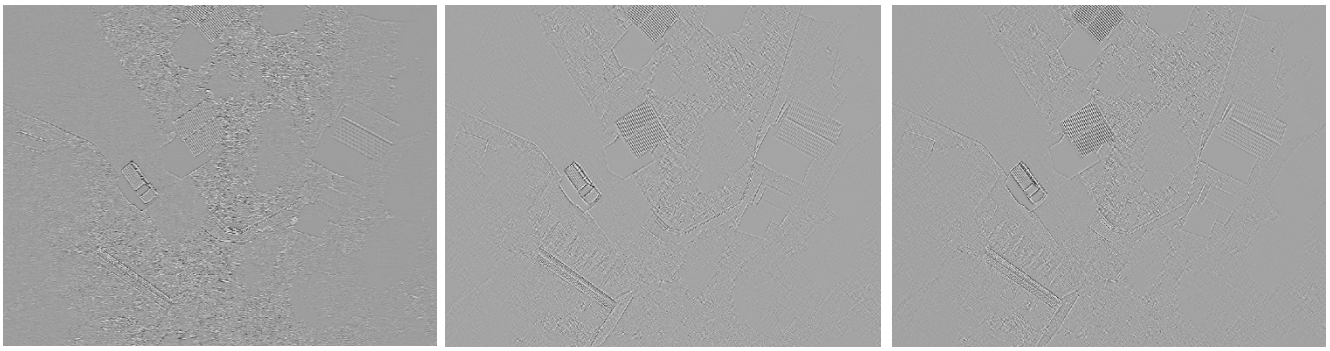
3.1.2. Spatial feature manipulation

In the 2nd group of the enhancement methods belong the image filtering techniques such as Spatial filtering; Convolution filtering; Edge enhancement; Directional first differencing, etc. In all these cases the "spatial frequency" of an image is changing by modifying the pixel values.

Filtering an image is a neighborhood operation procedure. In the output (filtered image) the value of any pixel is determined by applying specific algorithms to the nearest pixels (neighborhood) of the corresponding pixel of the input image. *High pass filters* (directional or edge detection) increase the spatial frequency and they do enhance the "edges" between non-homogeneous groups of image pixels. The purpose of edge enhancement is to highlight fine detail in an image or to restore, at least partially, details that have been blurred. Some different edge detection (sharpen) filters have been applied on the stretched image in **Figure 2** and the results are presenting in **Figure 3** and **Figure *4***. A combination of two (or more) filters can be applied on the same image, for example, **Figure 6**b and **Figure 6**d images have resulted from the implementation of the Sobel North East filter (**Figure *5***a) and the Morphological Filter - Erosion 3x3 and Erosion 7x7 respectively. The standard deviation 3x3 filter has been also applied to the input image and the result is presenting in **Figure 7.** In this case, the value of each output pixel is the standard deviation of the 3x3 neighborhood around the corresponding input pixel.

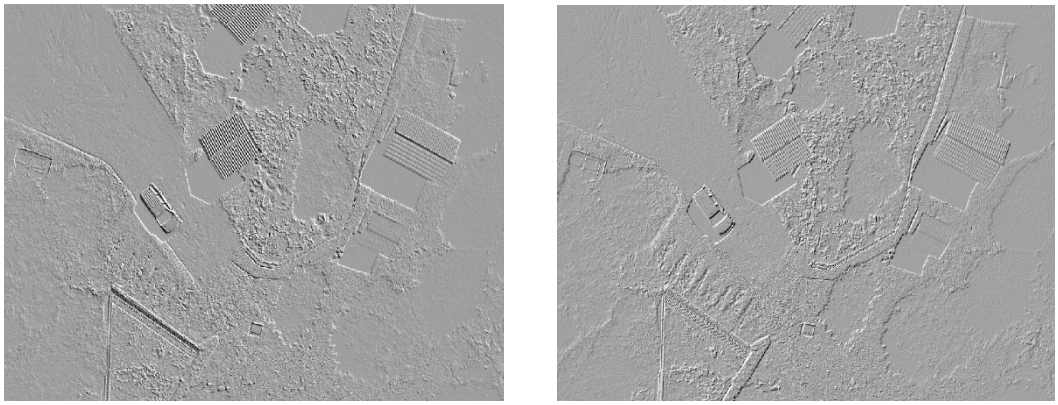
On the other hand, *Low pass filters* smooth the appearance of the image (smoothing filters). They are used for blurring (blurring is a pre-processing step for small details removal) and noise reduction in the image. An mxn linear spatial filter requires an mxn mask of coefficients. The coefficients are selected based on the type of the filter each time. Moreover, the size of the mask (m,n) also significantly affects the result.

Convolution is a linear filter algorithm (**Figure 9**) and is also a neighborhood operation. Each pixel of the output image is the weighted sum of surrounding (neighboring) input pixels. In the image convolving procedure, a kernel is needed. The kernel is the matrix of the weighted factors which slides across the image and is multiplied with the input pixels values. The result of this procedure is the enhancement of the input image, in a certain desirable way.



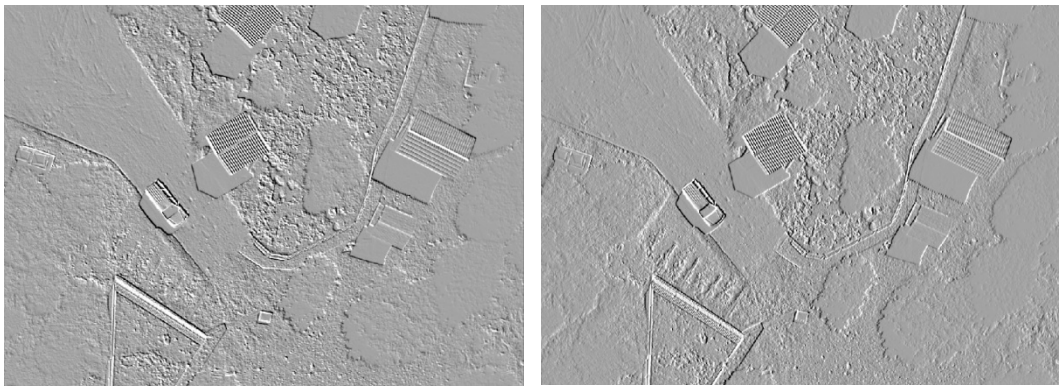
a. b. c.

Figure 3: *a. Horizontal Edges, b. right diagonal Edges, c. left diagonal Edges*



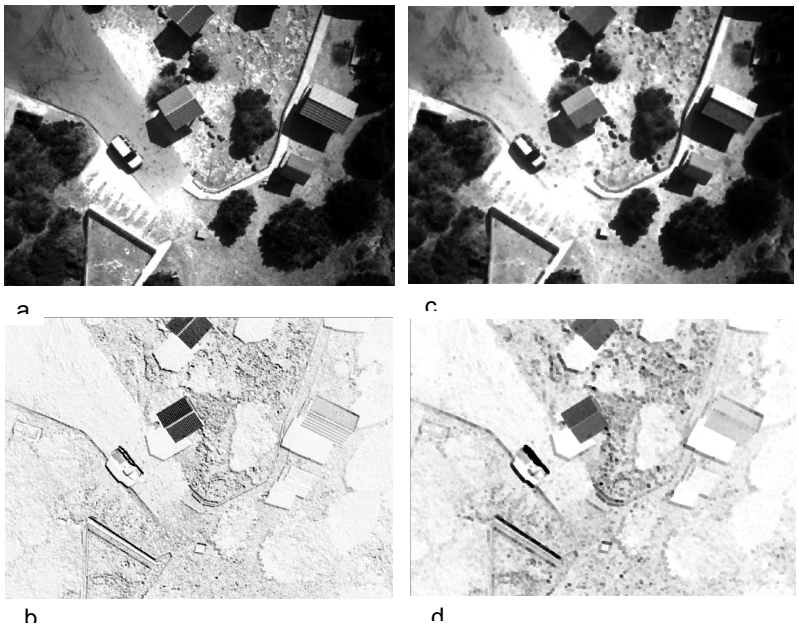
a. b.

Figure 4*: a. Roberts cross North East, b. Roberts cross North West*



a. b.

Figure 5*: a. Sobel North East, b. Sobel East*



a. c.

b. d.

Figure 6: ***a***. Morphological Filter - Erosion 3x3, **b**. Morphological Filter Erosion 7x7, **c**. Sobel North-East + Morphological Filter - Erosion 3x3, **d.** Sobel North-East + Morphological Filter Erosion 7x7

Figure 7*:* Standard deviation 3x3 filter applied to the input image.

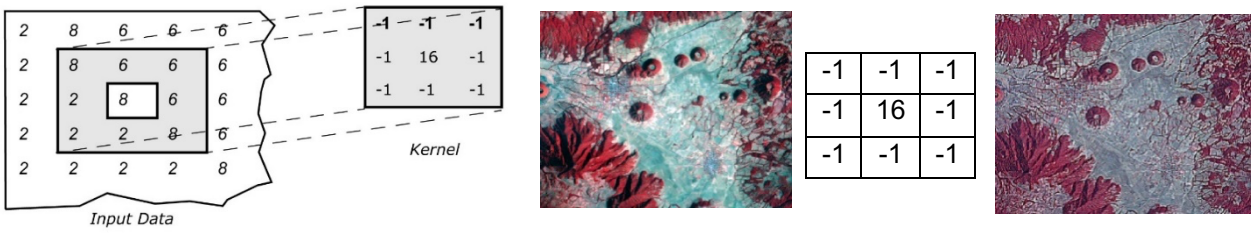


Figure 8: *Kernel is a mask used to apply filters to images. In this case, a* 3x3 high pass kernel is applying to the input image and the result is presenting on the right image *(Ludwig_ImageConvolution*).

$$V = \left| \frac{\sum_{i=1}^{q} \left(\sum_{j=1}^{q} f_{ij} d_{ij} \right)}{F} \right|$$

where:

$f_{ij} =$ the kernel coefficient at column *i*,row *j*

$d_{ij} =$ the pixel value at column *i*, row *j*

$q =$ the dimension of the kernel (i.e., 3X3)

$F =$ the sum of the kernel coefficients (if 0, then 1)

$V =$ the output pixel value

Figure 9: This formula gives to any pixel of the new image the new values based on the kernel coefficients.

3.1.3. Multi-image manipulation

The 3rd group of image enhancement methods refers to the different mathematical techniques which can be used to create, expand, transform, analyze or compress multiple bands of image and develop green vegetative index images (e.g., the NDVI). Some of these techniques are (i) the principal component analysis (PCA), (ii) the Tasseled cap, and (iii) the Indices.

i. PCA and Tasseled cap

Both PCA and Tasseled cap are useful when the input data is multiband imagery so these algorithms help to the simplification of the data. **Principal component analysis** transforms a multi-band image into a series of uncorrelated images ("components") that represent most of the information present in

the original dataset. **Tasseled cap transformation** transforms a multi-band image into a series of images optimized for vegetation studies using coefficients specific to a particular sensor.

ii. Indices

Indices are algorithms that create new images by mathematically combining the pixel values from multiple image bands. Common uses of indices are the reduction of radiometric differences, the minimization of the shadow effects, vegetation analysis, etc. Vegetation indices are generally based on the characteristic of healthy vegetation which is the low reflection of the plant in the visible spectrum and the high reflectivity in the infrared.

- ***Normalized Difference Vegetation Index (NDVI)***

NDVI is resulted by the ratio of the Red visible band and the near infrared band (***Figure 10***) and is widely used as a measure of both the presence and the health of vegetation. The NDVI values range from -1 to +1 and is positive when NIR > R. Larger NDVI values result from larger differences between the NIR and Red bands. NDVI is based upon findings that the chlorophyll in plant leaves strongly absorbs red visible light (from 0.6 to 0.7 μm), while the cell structure of the leaves strongly reflects near-infrared light (from 0.7 to 1.1 μm). Combining that with the fact that the healthier the plant, the higher the percentage of chlorophyll is contained in the plant, the high value of this index in vegetation studies, is becoming fully understood (**Figure 11**). In **Figure 12**, the NDVI algorithm has been applied to the same input image. In this case, the NDVI resulted after the embodiment of the NDVI algorithm in a py script. The health of the vegetation in this case can be observed by the intensity of the green color. The greener the color of the vegetation (NDVI values +0,75 to +1) the healthier and denser the vegetation.

$$NDVI_p = \frac{NIR_p - R_p}{NIR_p + R_p}$$

Figure 10*:* This relation gives the NDVI where NIR is the near-infrared response of pixel p; R is the visible red response of pixel p.

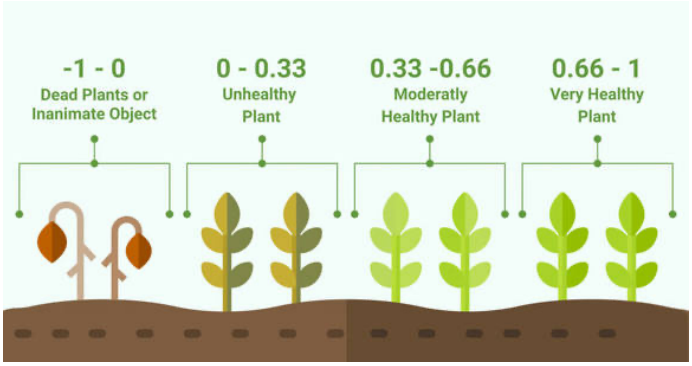


Figure 11: The NDVI values range from -1 to +1. The above image explains the relation between the NDVI values and the health of the plants (eos.com platform).

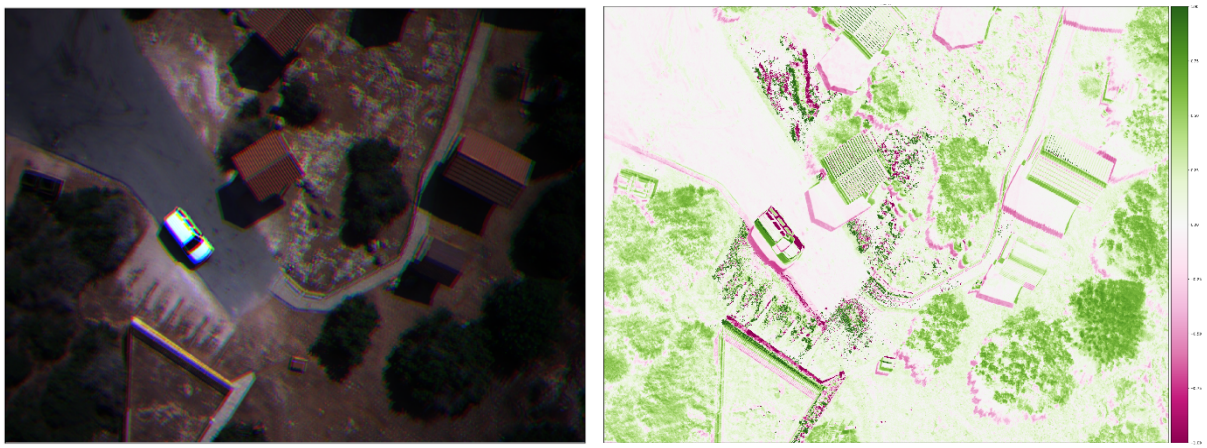


Figure 12: On the left is displayed the input multispectral image while on the right is presenting the NDVI image resulting after the implementation of the NDVI algorithm on the left image. The greener the trees on the NDVI image, the healthier they are.

- ***Burn Area Index (BAI)***

The BAI index is computed by the relation between RED and the NIR band (BAI= 1/(0.1-RED)^2+(0.06-NIR)^2) and it highlights burned land in between this spectral range. The index is computed from the spectral distance from each pixel to a reference spectral point, where recently burned areas converge. The presence of white (brighter) pixels in the output, indicates burned areas. Find in **Figure 13** an example of applying the BAI index on a Landsat8 image.

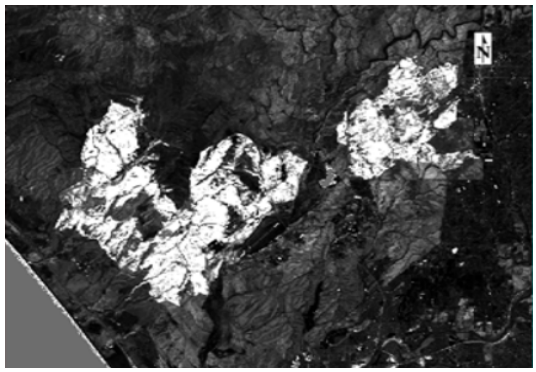


Figure 13*:* BAI index has been applied on a Landsat 8 image captured after a conflagration. The brighter pixels on this BAI image indicate burned areas.

- ***Global Environment Monitoring Index***

GEMI is a non-linear vegetation index like the NDVI but it is more sensitive to atmospheric effects. It is also calculated using the Red and NIR bands.
GEMI= (n(1−0.25n)−((RED−0.125)/(1−RED)) *Where n = (2 * (NIR ^2 - RED ^2) + 1.5 * NIR + 0.5 * RED) / (NIR + RED + 0.5).*

3.1.4. False color composite

An image can be visually displayed either by one band at a time (grayscale image) or by a combination of three bands (red, green, blue). By associating each band (not necessarily a visible band) to one of

the three colors, a color composite image is produced. This can be a true-color image or a false-color image. The false-color image results from the combination of the 3 colors with not the three visual primary color bands, so the result is not what would be observed by the human eyes. There are many possible schemes of producing false-color composite images and some of them are very useful for detecting certain objects in the images (e.g trees). One of the most common false-color composition is ***Red color ->NIR band, Green color -> Red Band, Blue color -> green band***. This composition allows vegetation to be detected readily in the image since the vegetation, in this case, appears in different shades of red (chlorophyll cause high reflection in the NIR band) depending on the condition (health, density) of the vegetation each time. In ***Figure 14***, the middle image shows the result from the application of this band combination on the input image on the left. The third image resulted after the implementation of the***Figure 1*** *linear stretch* on the middle image. This specific false-color composition is very useful for a better understanding of the stress, the density, and the presence of the vegetation.

Figure 14*:* On the left is the rgb image, in the middle is the result when the 4-3-2 band combination (when the 4rth band is the NIR) applied on the left image. The right image has been resulted after the linear stretch has been applied on the middle image.

3.2. Photogrammetric processing

The input image for all the above algorithm implementations is a part of a large dataset that has been selected aerial by a UAV platform. Specifically, 337 images were captured by a multispectral camera which was mounted on a UAV platform. The camera that was used, measures electromagnetic radiation in the blue, green, red, red edge, near infrared (NIR) and thermal infrared 8-14um bands. A sample of the captured images is presenting in ***Figure 15*** where the first row (IMG001_1:IMG002_6) refers to the 6 bands of the 1st image (image ID number=1) of the whole dataset. A procedure has been followed in order to stack the bands of each image together.

Figure 16 is presenting the result of the *band stacking procedure* where the 6 bands of the IMG_001 are stacked together and so, a multispectral image has been created. The appropriate flight planning parameters such as the overlap of the images, the altitude of the flight and the area boundaries were set, and the imagery was then collected. A series of algorithms run on this dataset in order to resolve the following tasks of detecting, selecting and matching points on the input dataset, estimating camera locations, generating depth maps, parameterizing texture atlas, blending textures, orthorectifying the images (***Figure 17***). The useful products that resulted from this procedure is an orthophoto and a 3D model. The orthophoto (***Figure 18***) is a photograph that has been digitally manipulated and has therefore

metric values. The important advantage of this model is that, when the user measures on it with a scale meter, he can recover all dimensions with the same precision as in a common diagram. By doing so, the geometric accuracy of the displayed objects can be ensured. The orthophoto is geometrically identical to a map. ***Figure 19*** is presenting the detailed 3D model of the same area. The 3D model represents terrain surface, sites, vegetation and other landscape elements in a three-dimensional scale. The 3D visualization expands the ability for analysis, exploration, presentation and management tasks related to the displayed area.

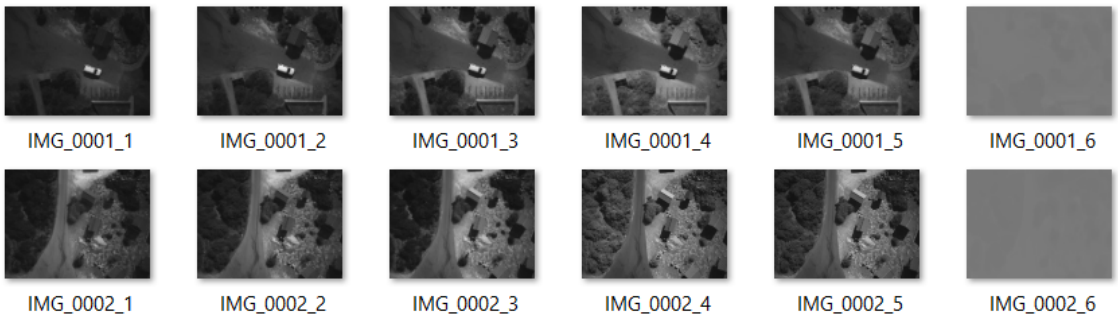


Figure 15: This image presents 2 images of the whole collected dataset where 001 and 002 refer to the image id numbers and the _1:_6 refer to the band numbering.

Figure 16: The example of the image stacking for the IMG_001. The 6 bands (blue, green, red, red edge, NIR, thermal) are stacked together and the result of this procedure is a multispectral image.

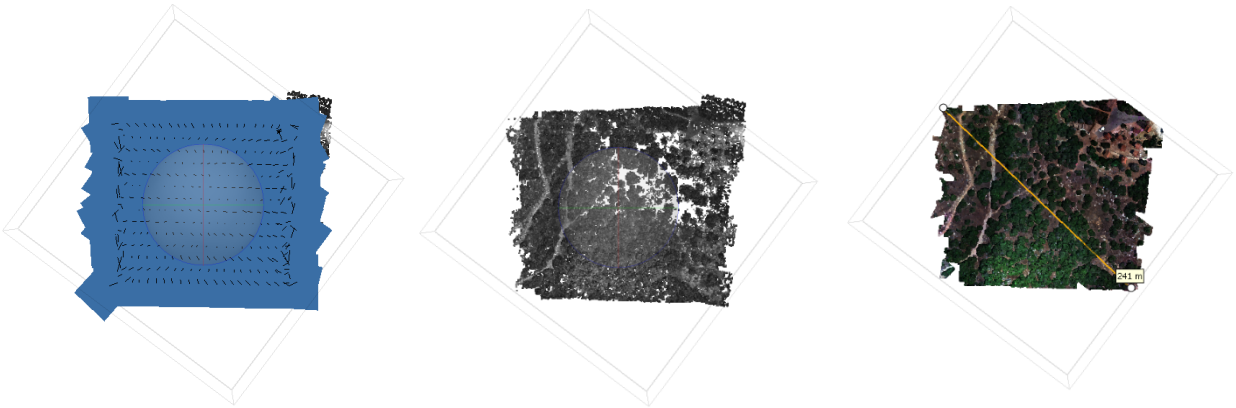


Figure 17*:* Photogrammetric procedure results. On the left image is presenting the result of the image alignment procedure where points from the images are detected, selected and matched, in order to be then correctly aligned. The sparse point cloud and the 3D model are presenting in the middle and right image respectively.

Figure 18: The orthophoto that resulted after the implementation of photogrammetric algorithms on a dataset of 337 UAV images.

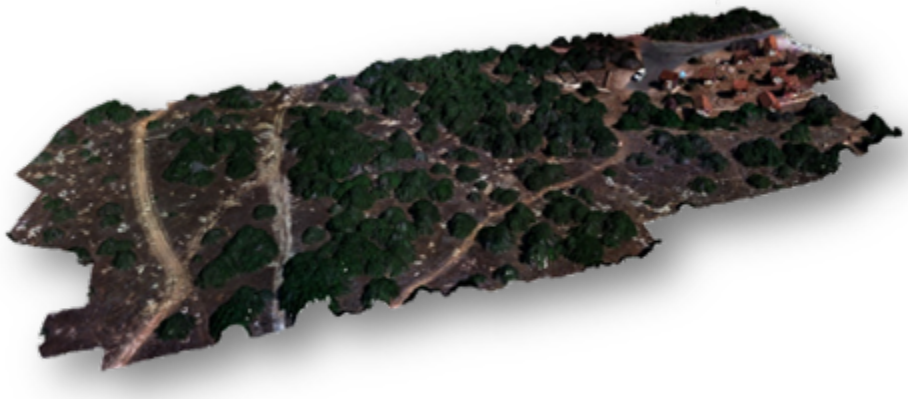


Figure 19: The 3D model which has been resulted after the implementation of different photogrammetric techniques on UAV imagery dataset.

3.3. Image classification

Image classification as well as object detection are computer vision problems. Image classification is the process of categorizing groups of pixels mainly based on their spectral characteristics. This procedure is characterized as supervised when some training data (small groups of pixels) are visually pre-categorized in order to create statistical measures to be applied to the entire image. One of the most common algorithms that is used in order to categorize the entire image based on the training data is the *maximum likelihood algorithm*. Maximum likelihood is a method for the estimation of the parameters of a probability distribution by maximizing the likelihood function. This function measures the goodness of fit of a statistical model to a sample dataset for the given values of the unknown parameters. The implementation of this algorithm on this computer vision problem means that the maximum likelihood uses the statistical characteristics of the training data (mean and standard deviation) and after that, it is computing the likelihood to each pixel to belong to individual classes. In the end, the pixels are grouping into the classes that show the highest likelihood. The above procedure as well as some smoothing post-processing algorithms have been applied on the orthophoto (**Figure *18***) and the result is presenting in **Figure 20**. The whole procedure has been performed in the ArcMapGIS platform.

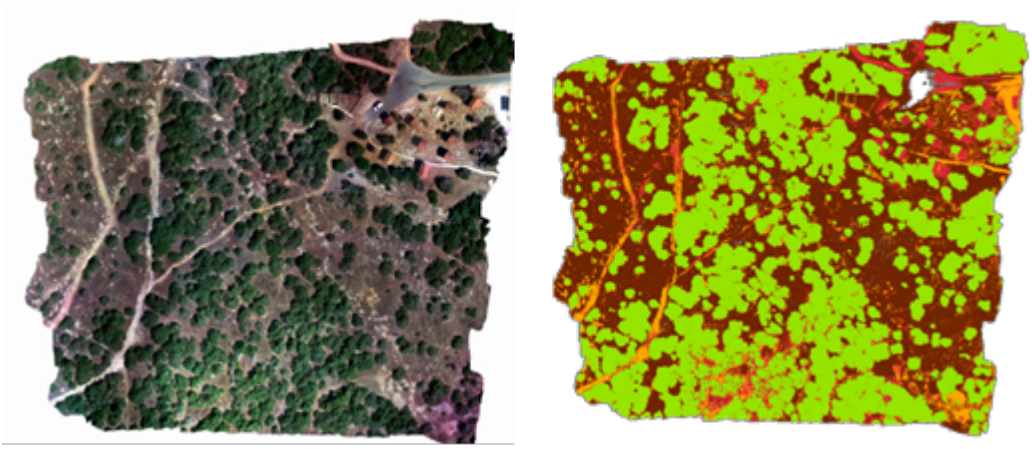


Figure 20 The orthophoto has been classified into 4 classes: trees (green); builted area (red); soil (brown); dirt roads (orange)

3.4. Machine learning

Machine Learning (ML) is an application of artificial intelligence (AI) that provides systems the ability to learn and improve based on experience. Such experience is known as training data so Machine learning algorithms build a model based on these data, in order to make predictions or decisions without being explicitly programmed to do so.

Hereafter we elaborate on object detection for detecting several objects of interest, such as people, vehicles etc., in real time using an aerial platform (i.e., a UAV). In order to achieve this, a Convolutional neural network [1], a class of deep neural networks, is used. Deep neural network (DNN) [2] is an artificial neural network with multiple layers between the input and output layers. There are

different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions. These components functioning similar to the human brains and can be trained like any other ML algorithm.

3.4.1. Object Detection

For the requirements of this project, several objects such as vehicles, people etc. need to be detected, and identified. Hence, as emphasized above a Convolutional Neural Network is used for object detection. Object detection is the computer technology that it is related to computer vision and image processing. It deals with detecting semantic objects of certain classes in images and videos.

One specific neural network is used, Tiny Yolo V4 [3], which is the latest version of the Yolo network models developed using Darknet framework [4]. Darknet is an open-source neural network framework developed using Cuda and the C programming languages. Tiny yolo v4 is mostly suited for real time scenarios due to having small number of layers. Hence, it has the ability to run object detection with higher framerates. It is trained using the Darknet framework on the COCO Dataset [5], which consists of 330K images, 1.5 million object instances of 80 different object types, including people, cars, trucks, busses.

A python application has been developed that utilizes the Tiny Yolo v4 trained model to detect and track objects in real time using the live camera streams of the UAVs. This application also utilizes the OpenCV library [6] and several tracking algorithms mention in [7] in order to track objects being the same in several sequential frames. The main algorithms used here are Kalman filtering and the Hungarian Algorithm, which uses Intersection over Union (IOU) score of the detected boxes over a sequence of frames. An example of IOU scores is shown in figure 21.

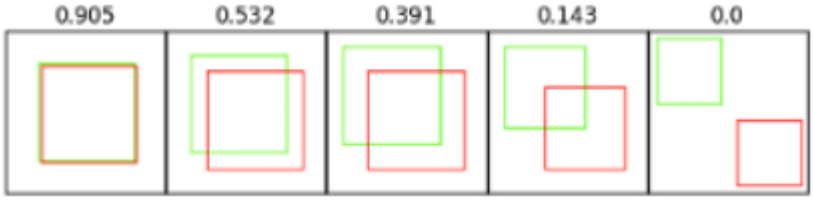


Figure 21 Intersection Over Union scores examples.

Kalman filtering is used for its prediction capabilities and much more precise tracking. Hence, for tracking all detections are firstly initialized with unique IDs starting from the first frame. Then, the detection boxes in the sequence of frames are compared against their predecessors for all tracked boxes using the IOU scores. If the score is higher than a certain threshold an identity matching has been made. Of course, the possibility that the same object does not have an overlapping detection in consecutive frame exists. This happens due to either missing frames from the stream or rapid movements of the UAV. In that case, the distance to all nearby objects is calculated and if the distance is less than a certain threshold, an association is made. Then, a Kalman predict is calculated so the next box comparison, on the next frame, will be made with a box slightly predicted (shifted) towards the direction that the detected object is moving, if it is indeed moving. A figure example of the algorithm is shown in figure 22.

Using this approach, we can identify specific objects, keep them in track and then represent them in real time. This is done by sending the tracked objects approximate GPS coordinates to the

platform using our API. The calculations of the GPS coordinates of the tracked objects are mentioned in section 3.4.3.

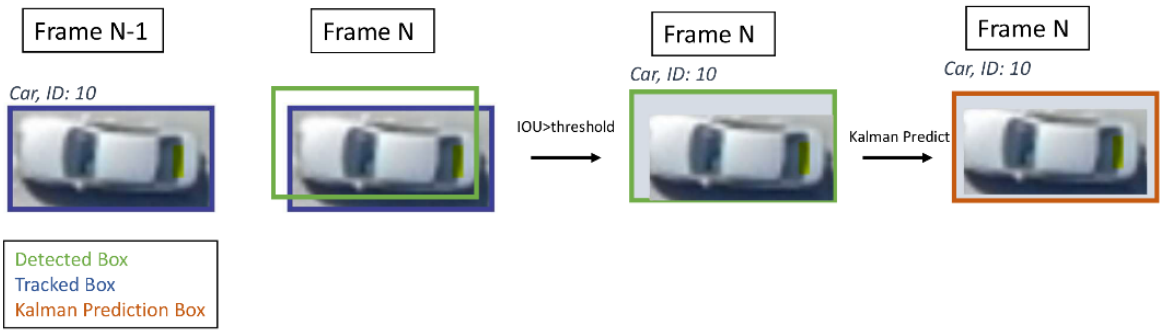


Figure 22 Tracking algorithm, identity matching association.

3.4.2. Scene Understanding

Deep learning algorithms such as Convolutional Neural Networks (CNNs) have been widely recognized as a prominent approach for many computer vision applications (image/video recognition, detection, and classification) and have shown remarkable results in many scene understanding applications 8,9,10. Hence, there are many benefits stemming from using deep learning techniques in emergency response and disaster management applications to retrieve critical information in a timely-fashion and enable better preparation and reaction during time-critical situations and support the decision-making processes 11. Even though CNNs have been increasingly successful at various classification tasks through transfer learning 12, their inference speed on embedded platforms such as those found on-board UAVs is hindered by the high computational cost that they incur and the model size of such networks is prohibitive from a memory perspective for such embedded devices 13,14. However, for many applications local embedded processing near the sensor is preferred over the cloud due to privacy and latency concerns, or operation in remote areas where there is limited or even no connectivity.

In the last decade, a lot of progress has been made on CNN-based classification systems. Numerous architectures have been proposed by the deep learning community fueled by the need to perform even better in image classification tasks such as the ImageNet Large Scale Visual Recognition Competition (ILSVRC). MobileNet 21 is a family of efficient networks that utilizes the idea of separable convolutions to offer reduced computational cost with slight degradation in classification accuracy. It applies a single filter at each input channel and then linearly combines them. Thus is designed can be easily parametrized and optimized for mobile applications. We utilize this backbone as the basis for training a network for image understanding in emergency response applications.

Significant relevant related work has looked into the the problem of aerial image classification for emergency response and disaster management. In 15 the authors propose a cloud based deep learning approach for fire detection with UAVs. The detection using a custom convolutional neural network (similar in structure to VGG16) which is trained to discriminate between fire and non-fire images of 128 × 128 resolution. The system works by transmitting the video footage from a UAV to a workstation with

an NVIDIA Titan Xp GPU where the algorithm is executed. Of course, in scenarios with limited connectivity missions there would be difficulties in applying this approach. Overall, the proposed approach achieves an accuracy in the range of 81 − 88% for this task. In 16 a method is proposed for detecting objects of interest in avalanche debris using the pre-trained inception Network for feature extraction and a linear Support Vector Machine for the classification. They also propose an image segmentation method as a preprocessing technique that is based on the fact that the object of interest is of a different color than the background in order to separate the image into regions using a sliding window. In addition, they apply post-processing to improve the decision of a classifier based on hidden Markov models. The application is executed on a desktop computer and not on an embedded device, with clock speed of 3GHz and 8GB RAM average a performance of 5.4 frames per second for 224 × 224 images. The accuracy was between 72 − 97%. Similarly, the work in 16 also targets fire detection with deep learning. Specifically, two pretrained convolutional neural networks are used and compared, namely VGG16 and Resnet50 as base architectures to train fire detection systems. The architectures are adapted by adding fully connected layers after the feature extraction to measure the classification accuracy. The different models average an accuracy of ∼ 91% for a custom database with an average processing time of 1:35 seconds on an NVIDIA GeForce GTX 820 GPU. The work in 18 proposes an approach comprised of a convolutional neural network called Fire_Net consisting of 15 layers with an architecture similar to the VGG16 network with 8 convolutional, 4 max-pooling, and 2 fully connected layers for recognizing fire in 128 × 128 resolution images. It is accompanied by a region proposal algorithm that extracts image regions from larger resolution images so that they can be classified by the neural network. The training of the system was performed on an NVIDIA GeForce 840M GPU, while the overall accuracy is ∼ 98% and the average performance is 24 frames-per-second on the particular GPU platform for 128 × 128 resolution images and without considering the overhead of the region proposal and region selection algorithms. In 19 a deep convolutional neural network is trained to classify aerial photos in one of 5 classes corresponding to natural disasters. The VGG network is used as the base feature extraction and a fully connected is placed on top of it to perform the transfer learning for the new task. An accuracy of 91% is achieved for a custom test set and on average less than 3 seconds are needed to process an image of 224 × 224 on an Intel Core i7 machine. From the literature analysis it is clear that existing approaches use existing pre-trained networks which adapt through transfer learning for the classification of a single event and primarily utilize desktop-class systems as the main computational platform that remotely process the UAV footage on GPUs. However, in certain scenarios the communication latency and connectivity issues may hinder the performance of such systems necessitating higher autonomy levels for the UAV and on-board processing capabilities. Also, the computing limitations of embedded platforms constitute the use of existing algorithms targeting desktop-class systems infeasible.

Algorithm implementation

Training a CNN for aerial image understanding for emergency response and disaster management applications first requires collecting a suitable dataset for this task. As such, a dedicated database for this task is constructed 20. The dataset construction involved manually collecting all images for four disaster events, 420 images of Fire/Smoke, 399 images for Flood, 401 images for Collapsed Building/Rubble, and 427 images for Traffic Incidents, as well as 1218 images for the Normal case.

Visually similar images such as for example active flames and smoke are grouped together. The aerial images of these disaster classes were collected from multiple sources such as the world-wide-web (e.g. google images, bing images, youtube, news agencies web sites, etc.), other databases of general aerial images. During the data collection process the various disaster events were captured with different resolutions and under various condition with regards to illumination and viewpoint. Finally, to replicate real world scenarios the dataset is imbalanced in the sense that it contains more images from the Normal class. Of course, this can make the training more challenging, however, a certain strategy is followed to combat this during training which will be detailed in the following sections. The operational conditions of the UAV may vary depending on the environment, as such it is important that the dataset does not contain only "clean" and "clear" images. In addition, data-collection can be time-consuming and expensive. Hence to further enhance the dataset a number random augmentations are probabilistically applied to each image prior to adding it to the batch for training. Specifically these are geometric transformations such as rotations, translations, horizontal axis mirroring, cropping and zooming, as well as image manipulations such as illumination changes, color shifting, blurring, sharpening, and shadowing. Each transformation is applied with a random probability which is set in such as a way to ensure that not all images in a training batch are transformed so that the network does not capture the augmentation properties as a characteristic of the dataset. The objective of all these transformations is to combat overfitting and increase the variability in the training size to achieve a higher generalization capability. Some samples from the dataset can be seen in Figure 23. Overall, with respect to the related works that consider multiclass problems almost 5× more data were collected. In addition, using augmentations the initial dataset was considerably expanded even further.

Figure 23: Example images from the Aerial Image Dataset for Emergency Response

Transfer Learning (Figure 24) is a method that uses transferable knowledge from another model and refines it using data available for the task at hand. The approach uses a base model that does very well at some general tasks like classifying all types of images into a thousand different classes. We do this by removing the last classifier layer, and fit a new classification layer which is then trained with new images. In this case, the base model simply functions as a 'fixed feature extractor' by creating a representation of an image that has captured generally relevant features.

For transfer learning a network more suited to embedded domains such as MobileNet 21 is selected. The feature extraction part is frozen for this network, applying all necessary preprocessing steps to the input image, and add a classification layer on top similar to prior works. In contrast to other works a global (per feature-map) average-pooling layer is applied prior to the dense layers followed by a softmax classification layer at the end. The average pooling reduces the parameter count and the subsequent computational and memory requirements and has shown to perform equally as well with the traditional approaches. Hence, the pre-trained models used for comparison are inherently more efficient in terms of memory and operations than the networks used in the literature for this task, which utilize fully connected layers.

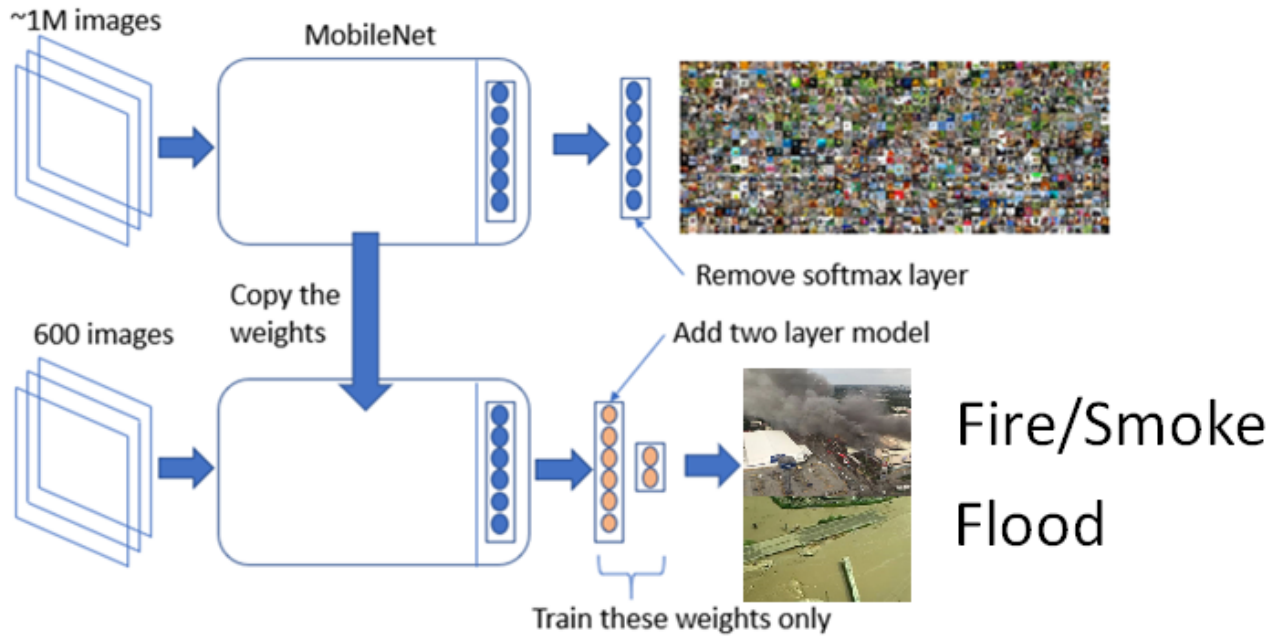


Figure 24: Illustration of the transfer learning concept used for training.

The first step in the training process is to split the dataset into training and validation. The bulk of the data are allocated to the training set and the rest is used for validation in a 0.85, 0.15 ratio. As mentioned prior, the Normal class is the majority class and thus is over-represented in the dataset. This reflects real-world conditions, however, if not addressed, it can potentially lead to problems where the network overfits and thus classifies everything as the majority class. To avoid issues due to the dataset imbalance the simultaneous use of majority class undersampling with oversampling of the minority classes within the same batch is performed. To do this we select the same number of images form each class to form a batch and this way all cases are equally represented. All the networks where trained

using a GeForce Titan Xp, on a PC with an Intel i7 − 7700K processor, and 32GB of RAM. The Adam optimization method was used for training with a learning rate of 0.001. The network is trained for 100 with a batch size of 32.

The main steps in this process are outlined below:

1) Take the trained MobileNet 21 model, a 28 Layer CNN model to classify images.
2) Truncate the softmax layer of this model and set the output of this model as last but one tensor representation of the image
3) Create a small model with two just layers — One Dense layer (with say 100 units) and another final softmax layer with as many units as the number of classes we want. Our first Dense layer must take the same input as the output of MobileNet.
4) Capture images and convert each image to its tensor using MobileNet. This transformed data is all that we need to train our model.
5) Using this training data, and our defined model, train our 2-layer model alone for a few epochs.
6) Create a joint model by attaching MobileNet and our 2-layer model.
7) Given a new image, run it through this joint model to generate a prediction.

Performance Evaluation

In this section, the analysis of the trained network is presented with results from the experimental evaluation. The per-class accuracy of the network is shown in Figure 25. Overall, we achieve an accuracy of over 90% for all classes. Figure 26 shows the confusion matrix for the model. As expected, the majority of mistakes are made from misclassifying some events as normal. In particular, misclassifying an image as traffic incident shows the highest error, which is to be expected since these types of incidents need a high level of context and encapsulate a large number of possibilities. In addition, the collapsed_building class has the lowest accuracy, this can be explained by the fact that these images generally have little structure and this makes it more difficult for the network to recognize such events. This can also be attributed to the fact that some collapsed_building images may also have other cues such as fumes/smoke, which makes them ambiguous. Another form of confusion is the fact that flood and road images have similar color cues. It is expected that incorporating a motion model or temporal information can help extract more context. Also, below we can see some classification results from the network.

CLASS	ACCURACY	# SAMPLES
Normal	0.92	183
Fire	0.95	63
Flood	0.93	60
Traffic Incident	0.92	65
Collapsed Building	0.90	61

Figure 25: Per class accuracy and number of samples

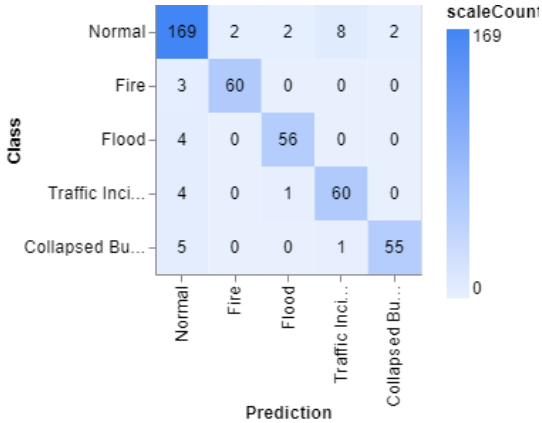


Figure 26: Confusion Matrix

Output

Normal	
Fire	
Flood	
Traffic Incide...	98%
Colla... Buildi...	

Output

Normal	
Fire	
Flood	30%
Traffic Incide...	70%
Colla... Buildi...	

Output

Normal	
Fire	100%
Flood	
Traffic Incide...	
Colla... Buildi...	

Output

Normal	
Fire	
Flood	
Traffic Incide...	100%
Colla... Buildi...	

Output

Normal	97%
Fire	
Flood	
Traffic Incide...	
Colla... Buildi...	

Output

Normal	
Fire	
Flood	
Traffic Incide...	100%
Colla... Buildi...	

Output

Normal	100%
Fire	
Flood	
Traffic Incide...	
Colla... Buildi...	

Output

Normal	
Fire	
Flood	100%
Traffic Incide...	
Colla... Buildi...	

Output

Normal	
Fire	
Flood	
Traffic Incide...	
Colla... Buildi...	100%

Output

Normal	
Fire	
Flood	
Traffic Incide...	
Colla... Buildi...	100%

Output

Normal	
Fire	100%
Flood	
Traffic Incide...	
Colla... Buildi...	

Output

Normal	
Fire	100%
Flood	
Traffic Incide...	
Colla... Buildi...	

3.4.3. Distant Metrics

Upon detecting and matching the objects, the ground distance of the objects from the sensor (e.g. camera) can be calculated using triangulation, as well as the lateral, longitude coordinates of the objects. For instance, the lateral and longitude coordinates are calculated along great-circle distance and bearing from the UAV's GPS coordinates as a start point.

To do that, initially some parameters taken from the UAV are required. Through the API, the relevant parameters are extracted including the altitude, lateral and longitude coordinates, the camera's angle and bearing of the UAV during the current timestamp. The following calculations are implemented for every object detection.

$$\psi = \theta + \left(\frac{H}{2} - j\right) \times \left(\frac{FOV_V}{H}\right) \qquad \phi = \left(i - \frac{W}{2}\right) \times \left(\frac{FOV_H}{W}\right) \qquad Y = h \times \tan(\psi)$$

The ψ is the angle between the detected object, the camera and the ground. φ is the angle between the object, the ground and the center of the image. Θ is the angle of the camera looking at the ground. H, W is the image's Height and Width. FOV v/h respectively are the vertical and horizontal field of view of the UAV's camera. The object's location on the image is represented with (i,j). The altitude of the drone is the h and the Y is the final ground distance calculated, from the UAV's location to the detected object. Upon calculating the ground distance, the object's lateral and longitude position can be calculated using the following equations.

$$Lat = asin\left(sin(lat_s) \times cos\left(\frac{Y_k}{ER_k}\right) + cos(lat_s) \times sin\left(\frac{Y_k}{ER_k}\right) \times cos(B)\right)$$

$$Lon = \left((lon_s + asin\left(\frac{sin(B) \times sin\left(\frac{Y_k}{ER_k}\right)}{cos(lat_s)}\right) + \pi) modulo (2 \times \pi)\right) - \pi$$

Lat_s, Lon_s represent the UAV's latitude and longitude, B is the bearing to the north. Yk and ERk represent the ground distance previously calculated and the Earth radius respectively in kilometers. Noted that all angles (lat,lon,bearing) are changed to radians in order to be used in these equations. Afterwards, lateral and longitude values are changed back to degrees. An example of results can be seen in Figure 27, in which the location of the detected objects is represented in the platform. Also, a popup window of the actual stream including the detections can be seen.

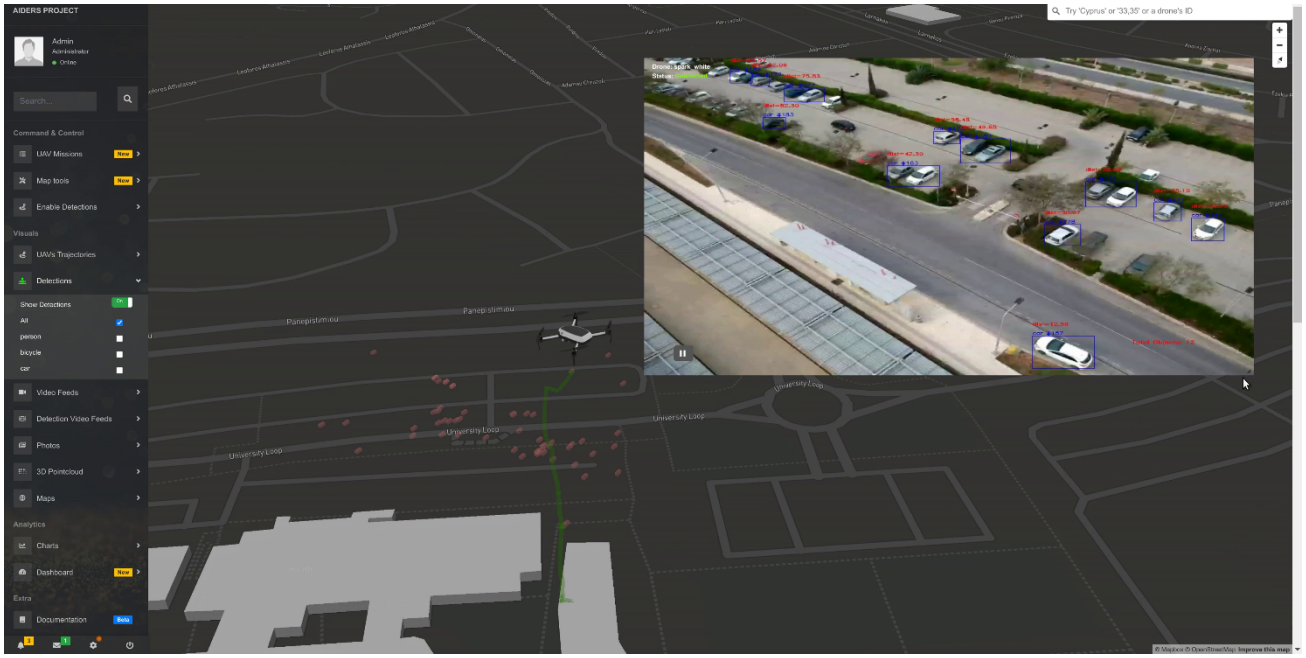


Figure 27 Detecting and positioning vehicles on the platform in real time.

3.5. Dynamical Modelling

In this part, we provide details regarding dynamical modeling algorithms for simulating/predicting forest fire spreading and flood spreading. Moreover, we present algorithms that enable UAVs to track both forest fires and floods, providing real-time information to first responders such as the specific disaster location, fire fronts growth, rate of spreading, etc., that they can use to manage more effectively the disaster situation.

3.5.1. Forest fire prediction (FARSITE)

Wildfire simulation has attracted significant research efforts over the years due to the potential in predicting wildfire spreading. This property has considerable importance for planning purposes and disaster management, allowing effective and constructive decision making.

The core model of fire spreading propagation was developed over the years from various researches. Initially, the equations to mathematically calculate the rate of speed and intensity of fire spread created by Rothermel[1]. Then, the elliptical model to estimate fire fronts growth was introduced by Richards[2]. Finally, building on top of the previous research, Finney et al.[3] developed the fire growth model called

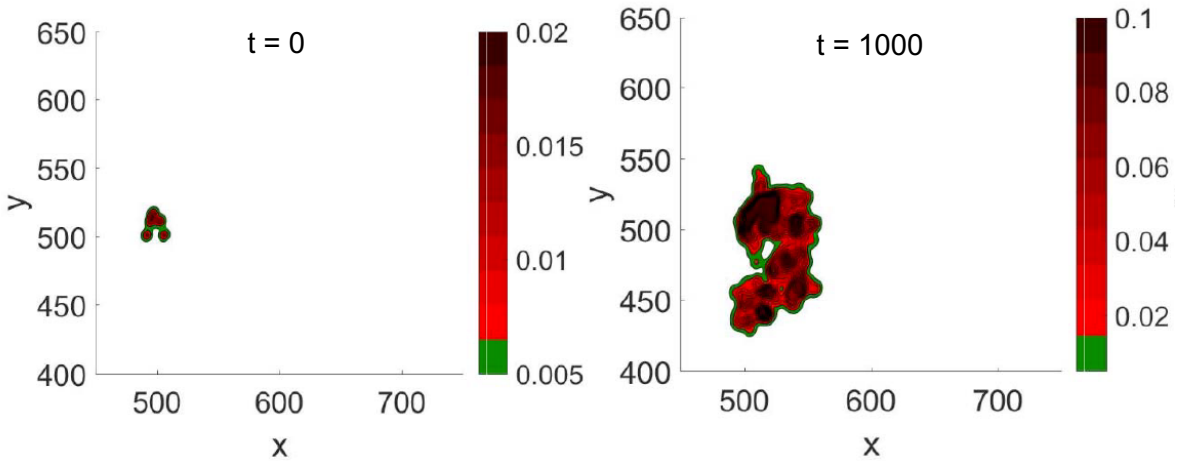
[1] R. C. Rothermel, A Mathematical Model for Predicting Fire Spread in Wildland Fuels. Ogden, UT, USA: U.S. Dept. Agric., 1972.
[2] G. D. Richards, “An elliptical growth model of forest fire fronts and its numerical solution,” Int. J. Numer. Methods Eng., vol. 30, no. 6, pp. 1163–1179, 1990.
[3] M. A. Finney et al., FARSITE: Fire Area Simulator: Model Development and Evaluation. Ogden, UT, USA: U.S. Dept. Agric., Forest Service, Rocky Mountain Res. Station, 2004.

Fire Area Simulator (FARSITE), which is considered the most reliable model and is widely used by several federal and state land management agencies to simulate the spread of wildfires.

FARSITE automatically computes wildfire growth and behavior for extended time periods, using information regarding geography, topography, terrain, fuel, and weather conditions. The modeling approach uses a vector propagation technique to calculate fire perimeter expansion (fire front) based on both space and time resolution of fire growth over the landscape. Specifically, FARSITE considers each point on the fire front as a new source of fire generation. The fire front is then propagated as a continuously expanding fire polygon or ellipse based on the weather condition (wind speed and direction), rate of fire spreading according to the vegetation type, and location geography. The model outputs vector fire perimeters (polygons) at specified time intervals that illustrate the fire propagation in the area under study, including information for the fire's spread rate and intensity. Fig. 28 presents simulation results using FARSITE, showing the fire evolution and intensity at different time steps.

FARSITE produces complex patterns of fire growth and behavior as result of spatial and temporal input data dependencies to the model. Specifically, the fire growth prediction model requires spatial data that consists of the fuels, weather, and topographic elements of fire behavior. For simulation purposes, weather and winds are usually input streams of data including windspeed, direction and cloud cover, whereas fuels and topography are provided as GIS data for easy access including elevation, slope, surface fuel, canopy cover, and crown details. For quick data generation, a simplified model of FARSITE that describe the fire front growth is possible by assuming uniform or gaussian distributions for fuel, weather and landscape. In case of uniform fuel distribution, uniform landscape and weather, and constant wind direction, fire fronts propagate in an elliptical fashion using FARSITE.

For practical applications, such as real-time in field fire growth projections, the weather and wind data can be extracted from weather recording stations that are very close to the fire location or using sensors in the field (e.g., on top of UAVs). All other data (fuel, landscape information) must be available for the specified location in a database for quick and easy access. In case that this is not possible, assumptions can be made along with real-time images from the fire that can help derive prediction regarding the fire spread.



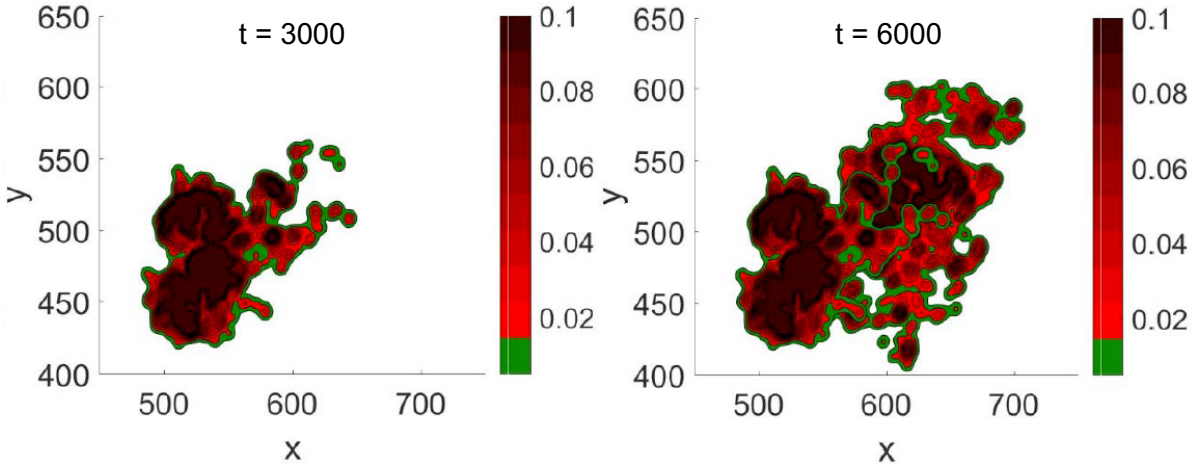


Figure 28: Fire spreading at different time steps using FARSITE. At t=0, the fire starts with a few sources around (500, 500) and then grows bigger and spreads around the field with the results shown at t = 1000, t = 3000, and t = 6000. The color bar indicates the fire's intensity level, where the darker the color, the higher the intensity.

FARSITE is a state-of-the-art fire propagation system that adopts the vector approach to fire growth modeling that enables the simulation of wind-driven wildland fires using the current fire behavior models. The simulations illustrated that time-and-space dependent fire behavior can be simulated with realistic consequences to spatial patterns of fire growth and behavior.

3.5.2. Flood prediction

Advances in flood modelling have evolved rapidly in recent years and are now at the center of various studies for mitigating flooding hazards. Flood modelling generally involves designing techniques and algorithms that describe flooding based on floodwater velocity, depth, and extent. Since flood water is a form of wave phenomenon that propagates in a downgradient direction, associating flow rate and water level changes, the models to describe this behavior are often designed to solve numerical expressions that estimate floodwater propagation from one point to another within a spatial domain.

Significant progress has been made in developing various flooding models, including one-dimensional, two-dimensional, and three-dimensional. Most preferable are currently the two-dimensional flood models. They provide more accurate results than the simplistic, one-dimensional models while requiring less computational time and thus faster than the three-dimensional models. The two-dimensional flood models such as TUTFLOW, SOBEK, and MIKE 21 solve the two-dimensional shallow water equation (SWEs) by utilizing appropriate numerical methods. These models have been benefited tremendously from the advances in remote sensing technology, such as airborne LiDAR and Synthetic Aperture Radar (SAR) data. Specifically, high-resolution topographic data provide accurate and high-quality input data in two-dimensional flood modelling with a detailed description of urban geometry such as streets, roads, and buildings. By having high quality input data available, the two-dimensional flood models can simulate urban flooding with comprehensive representation of flow hydrodynamics along with small scale topographic features that enable less uncertainty to the generated results.

Optimal flood models provide the friction, slope, acceleration, gravity, mass, and momentum of the water during a flooding situation. However, the estimation of all those parameters depends on the model accuracy and input data availability. In case of low-quality or less available data, flood models with simpler mathematical representations can be applied, including simplified two-dimensional models or reduced complexity models (RCM) such as the LISFLOOD-FP, JFLOW, and ISIS-FAST. Along with coupled 1D/2D models and Cellular Automata (CA) based models, RCM can solve the kinematic wave, diffusive wave, and inertial wave equations which result from various simplifications of the SWEs.

Overall, the use of two-dimensional flood models received positive momentum for flood prediction from various sources and are considered optimal in flood modelling. These models have so far provided realistic applications and demonstrated high performance in the areas of urban flood modelling.

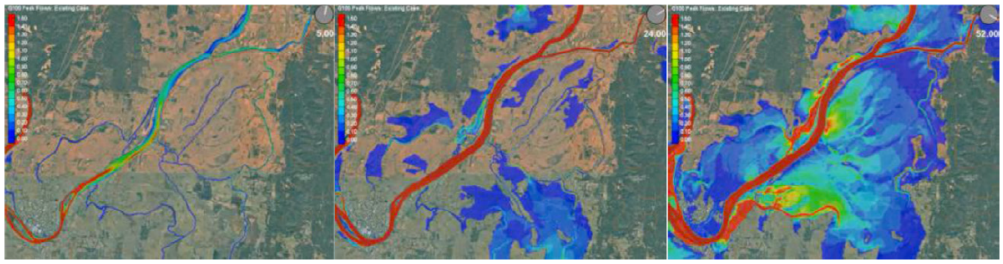


Figure 29: Flood prediction using TUFLOW modeling solution.

3.5.3. Tracking

Fire and flood prediction models are exceptional tools for planning and disaster management. However, during a wildfire or flood, the situation changes very fast due to various conditions that are not easily predictable by the models. Thus, real-time tracking and monitoring of the unfolding disaster is crucial for assisting humans during hazardous tasks. The use of manned helicopters for fire and flood tracking tasks is quite common method, as well as the continuous inspection of satellite images. Compared to both these methods, the use of UAVs (drones) significant advantages for tracking tasks since they conserve sizable operation costs and minimize the risk compared to the use of manned helicopters. Furthermore, UAVs are available on-demand and can provide real-time coverage compared to the satellite coverage that is periodic with unfavorable revisit times.

The characteristics of UAVs make them very favorable for the task of tracking. UAVs can fly at low altitudes and capture high-resolution imagery and broadcast frequent updates to disaster management crews. However, since the disaster area from fire or flood can become large is necessary to deploy a team of drones to achieve good coverage and frequent updates. Tracking a wildfire or flood poses various challenges and requires the design and utilization of complex algorithms for autonomous control, coordination, cooperation, and data exchange between a team of UAVs. Fortunately, various such algorithms have been developed that enable a team of UAVs to perform perimeter tracking, disaster coverage, and disaster frontier tracking.

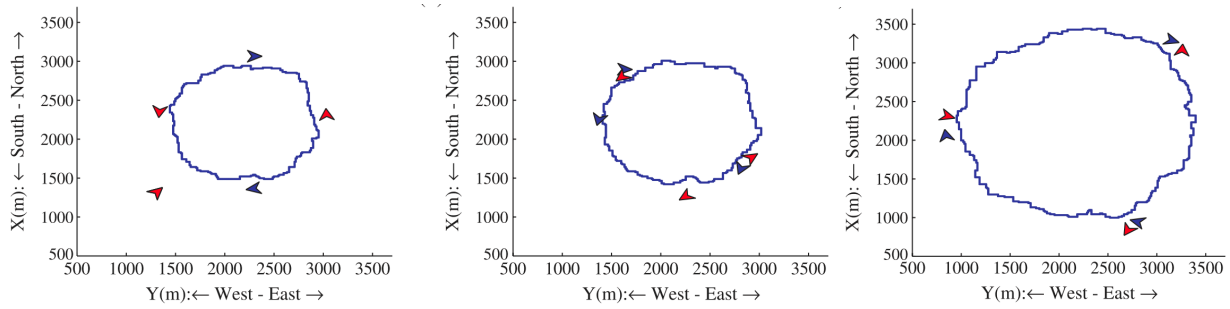


Figure 30: Perimeter tracking by a team of UAVs as it grows. (a) four UAVs monitor the perimeter, and another one is approaching. (b) four UAVs reach the checkpoints for data exchange, and the other two monitor the perimeter. (c) six UAVs monitor the perimeter and reach the checkpoints for data exchange.

Perimeter tracking algorithms enable a team of UAVs with appropriate hardware (e.g., camera, IR sensors, etc.) to detect the edge of fire or flood. Then they can fly to a safe altitude close to the boundary of the disaster and continuously calculate the coordinates as it grows over time. However, the perimeter can grow significantly, and new UAVs need to be added to avoid data transmission delays. Figure 30 shows an example of a perimeter tracking algorithm. Initially, in Figure 30(a), four UAVs monitor the perimeter, and another is approaching. As can be seen, the two blue UAVs fly clockwise while the two red UAVs counterclockwise. This formation allows splitting the distance of the perimeter with each UAV monitoring the same perimeter length. In Figure 3.3.2(b), as the perimeter grows, two more UAVs are added to the team, while the four “oldest” UAVs reach a checkpoint for data exchange (i.e., exchange of perimeter GPS coordinates) that will propagate through the UAVs and finally reach the base station from the closest UAV. Finally, in Figure 3.3.2(c), the algorithm controls the six UAVs to monitor the same length of the perimeter to minimize the data delay through the UAVs so that the base station is updated with the latest available perimeter GPS coordinates.

Disaster coverage algorithms coordinate a team of UAVs to continuously fly above the fire burning or flooded area and collect data such as images for analysis, check for survivors, etc. For the coverage task, the UAVs are controlled dynamically to maximize their coverage field while satisfying certain constraints such as maximum distance for successful communication, minimum distance for collision avoidance, and minimum altitude for a safe distance from the ground. Figure. 31 presents a screenshot from a coverage mission of a team of UAVs during a wildfire. The UAVs are equipped with cameras that can identify the intensity of the fire. Based on this information, the UAVs control their position and their altitude, adjusting their field of view (FOV) and the quality of images while exchange data between them to maximize the wildfire coverage as it evolves.

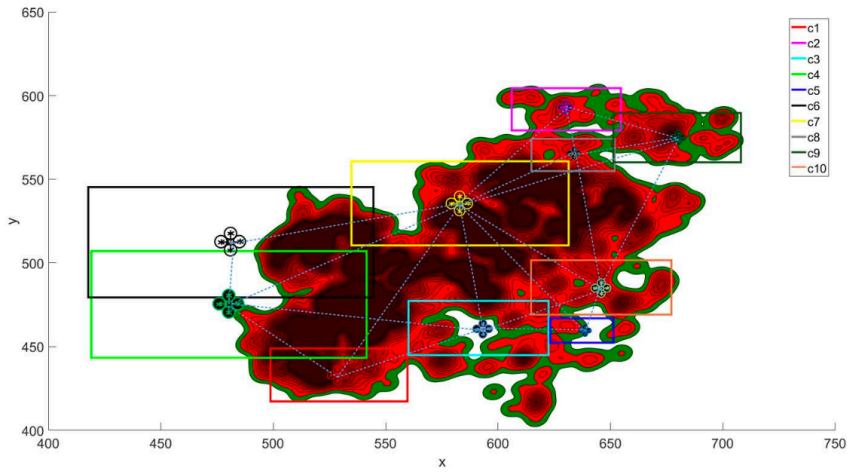


Figure 31: Shows the positions of a team of 10 UAVs and their field of view (FOV) during a wildfire coverage mission.

The disaster frontiers, i.e., the locations that the disaster moves forward (such as fire fronts), are critical since it allows the disaster management teams to take necessary actions that can potentially reduce the extent of the disaster. Thus, algorithms have been developed that enable a team of UAVs to track the disaster front. Specifically, during the disaster front tracking mission, the UAVs will approach and detect the fire or the flood boundary, similar to the perimeter tracking task. They will then determine the points on the boundary that move faster, allowing them to identify the disaster's frontier. After that, the UAVs will fly above the frontier, allowing close inspection and real-time monitoring by the disaster management teams. Figure 32 illustrates the result of disaster frontier tracking algorithm controlling a team of UAVs at different time steps.

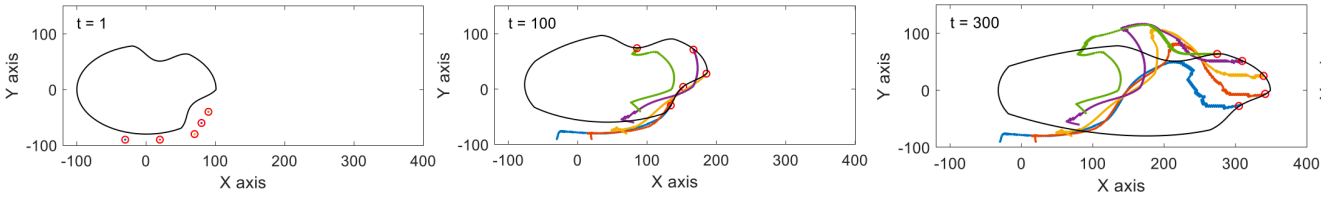


Figure 32: UAVs track the frontier of a disaster as it grows.

4 Conclusion

In this deliverable, we have first reviewed the state-of-the-art algorithmic tools for data analysis that can best address the user requirements in emergency response information. In addition, through real examples we have demonstrated the applicability of these algorithms in extracting the relevant data and discussed how this information can be depicted on the AIDERS ground control platform for visualization and decision support.

5 References

1. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press (pp. 326-366)..
2. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press.
3. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.
4. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
5. Lin, T. Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... & Dollar, P. (2019). Microsoft COCO: common objects in context (2015). *arXiv preprint arXiv:1405.0312*.
6. Beyeler, M. (2017). *Machine Learning for OpenCV*. Packt Publishing Ltd.
7. Makrigiorgis, R., Kolios, P., Timotheou, S., Theocharides, T., & Panayiotou, C. G. (2020, May). Extracting the fundamental diagram from aerial footage. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)* (pp. 1-5). IEEE.
8. E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657, Feb 2017.
9. Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 2018.
10. G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns. *IEEE Transactions on Geoscience and Remote Sensing*, 56(5):2811–2821, May 2018
11. Tien Dat Nguyen, Shafiq R. Joty, Muhammad Imran, Hassan Sajjad, and Prasenjit Mitra. Applications of online deep learning for crisis response using social media information. *CoRR*, abs/1610.01030, 2016.
12. Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.
13. Y. Wang, Z. Quan, J. Li, Y. Han, H. Li, and X. Li. A retrospective evaluation of energy-efficient object detection solutions on embedded devices. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 709– 714, March 2018.

14. Muhammad Shafique, Theo Theocharides, Christos Bouganis, Muhammad Abdullah Hanif, Faiq Khalid, Rehan Hafiz, and Semeen Rehman. An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 03 2018
15. S. Kim, W. Lee, Y. s. Park, H. W. Lee, and Y. T. Lee. Forest fire monitoring system based on aerial image. In *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–6, Dec 2016.
16. Mesay Belete Bejiga, Abdallah Zeggada, Abdelhamid Nouffidj, and Farid Melgani. A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. *Remote Sensing*, 9(2), 2017.
17. Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin, and Jahn Thomas Fidje. Deep convolutional neural networks for fire detection in images. In Giacomo Boracchi, Lazaros Iliadis, Chrisina Jayne, and Aristidis Likas, editors, *Engineering Applications of Neural Networks*, pages 183–193, Cham, 2017. Springer International Publishing.
18. Yi Zhao, Jiale Ma, Xiaohui Li, and Jie Zhang. Saliency detection and deep learning-based wildfire identification in uav imagery. *Sensors*, 18(3), 2018.
19. Andreas Kamilaris and Francesc X. Prenafeta-Bold. Disaster monitoring using unmanned aerial vehicles and deep learning. In *Disaster Management for Resilience and Public Safety Workshop, in Proc. of EnviroInfo2017*, Luxembourg, September 2017.
20. Christos Kyrkou and Theocharis Theocharides, "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 1687-1699, 2020.
21. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.