AI: Lab - Computer Vision and NLP

# Fire Detection Through Normal RGB Videos for Forest Fire Prevention

Project Report

Alessio Olivieri
Emilio Soriano Chávez
Robert Li

## 1 :: Introduction

Forest fires represent an important threat to natural ecosystems, wildlife, and human lives. Early detection is essential to prevent extensive damage and reduce risks associated with them. Conventional fire detection systems primarily rely on *smoke* or *temperature-based* sensors, which have been used for decades. Late approaches have also involved watchtowers that rely on a person to manually file a report if a fire is spotted. However, these approaches have inherent limitations that restrict their effectiveness, particularly in outdoor environments and forested areas.

*Smoke-based* fire detection, while effective in enclosed spaces, such as buildings, is not as effective in open environments due to smoke dispersion, wind patterns, and smoke densities. Moreover, false alarms are frequent when smoke from non-fire-related sources, such as industrial activities or vehicle emissions, trigger the sensors.

*Temperature-based* fire detection can easily identify regions with elevated temperatures, but this approach cannot differentiate between regular heat sources and actual fire incidents by itself. This limitation reduces its ability to provide early warnings, particularly in complex outdoor settings where temperature variations are common.

*Watchtowers* require an individual to always be present, which can be costly depending on how large of an area is to be covered.

Our approach attempts to combine the feed obtained from traditional video or surveillance cameras with a motion detection algorithm and a model to predict forest fires in real-time. This approach is inherently more cost-effective and requires less human intervention than the aforementioned methods.

## 2 :: Background

Fire and smoke detection in forests has largely been a complex problem. As stated in the New Scientist Magazine, “there are too many cameras and too few pairs of eyes to keep track of them” [#]. This represents a real need for intelligent video content analysis to support camera operators. Although it can be relatively easy for human beings to detect whether there is a fire or not, it is not easy to replicate human intelligence.

One of the ways to approach the problem is a fire detection technique that focuses on using Infrared (IR) Cameras. However, the high cost of infrared cameras compared to conventional digital cameras shooting in the visible range represents a problem. Another issue that occurs when using an IR-based method is the fact that smoke spreads faster than fire, and in most cases will occur much faster in the field of view of the cameras. In wildfire applications, it may not be even possible to observe flames for a long time. Additionally, "wildfire smoke detection using an ordinary IR camera with a spectral range of 8–12 μm is very difficult as smoke is invisible" [1]. This is demonstrated in Figures 1 and 2, which show tests using both IR and visible range cameras over the same area.

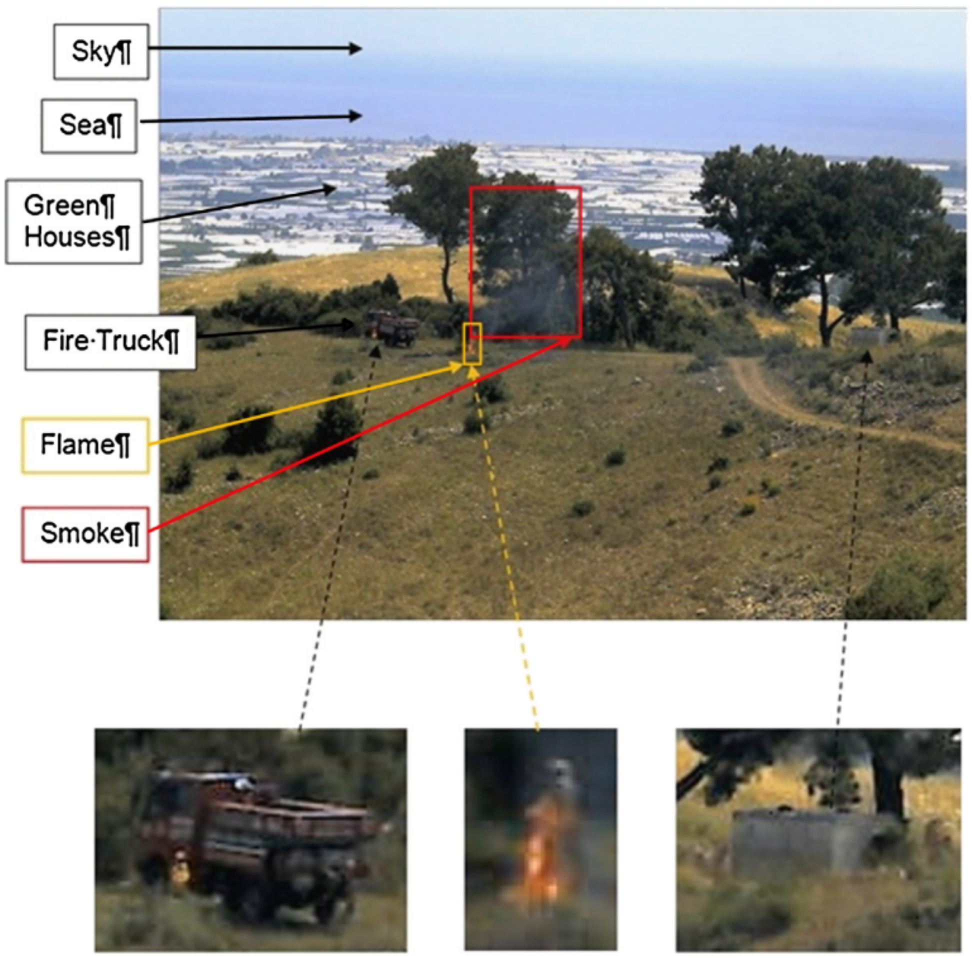

Figure 1 | **Wildfire Smoke and Fire Captured using a Visible-Range Camera** [1]

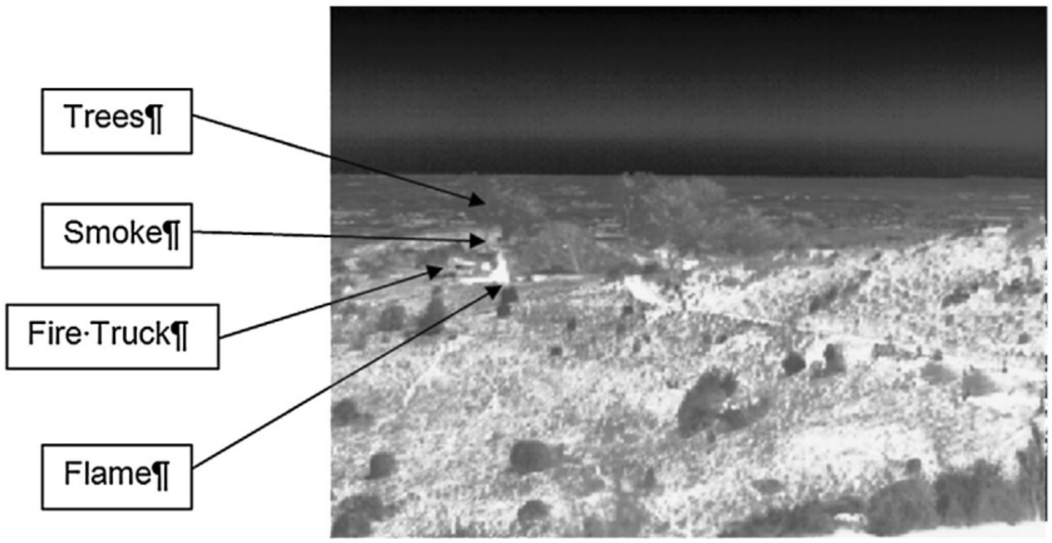

Figure 2 **| Wildfire Smoke and Fire Captured using an IR Camera** [1]

The approach we propose is a technique that focuses on the *color* and shape characteristics of smoke and fire. However, due to the variability of shape, motion, transparency, colors, and patterns of smoke and flames, many of the existing Video Fire Detection approaches are still vulnerable to false alarms, and although the system can help operators recognize smoke and fire, it can still not reliably work without an operator.

---

## 3 :: Implementation

### 3.1 - Motion Detection Algorithm Design

The algorithm for detecting smoke or fire in frames using cameras that shoot video in the visible spectrum can be divided into the following components:

1. Motion detection using the *background subtraction method* with recursive update of thresholds and estimated background.
2. Computation of regions with potential motion to be analyzed with a model.
3. Detecting the presence or absence of fire/smoke in the regions using a model.

Motion detection is done by a method known as *background subtraction*. The basic concept behind this approach is to compare consecutive frames of a video and identify pixels or regions that exhibit significant changes. These changes can then be attributed to moving objects against a relatively stationary background.

In the specific case of forests, where the camera is located in an open area, the camera will pick up changes in brightness due to the sun, changes in weather conditions, among others. Therefore, an algorithm is needed that takes this feature into account and adapts to changes in the environment.

In order to approach the aforementioned problem, we designed an algorithm that generates an average background and updates the thresholds for each pixel. If the change in the value of a current frame's pixel compared to an averaged background's pixel is greater than the threshold, we conclude that the pixel is *moving*. It is necessary to set limits for each pixel, as the change in lighting can be uneven. This can be observed in Figure 3.

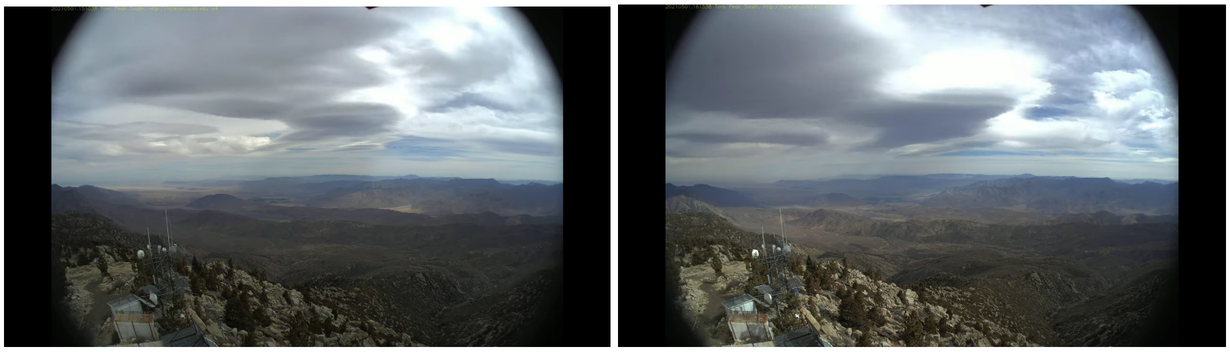

Figure 3 **| Uneven Lightning Changes**

Changes in the brightness and illumination of the area associated with clouds, the illumination changes unevenly, so the rocks on the left side remain dark.

After identifying the *moving* pixels, we need to distinguish actual motion from noise or minor temporal changes. We do this by creating a mesh grid and computing the amount of *moving* pixels inside the grid. If the area these pixels occupy is big enough we determine that there is movement in the area. The procedure is shown in Figure 4.

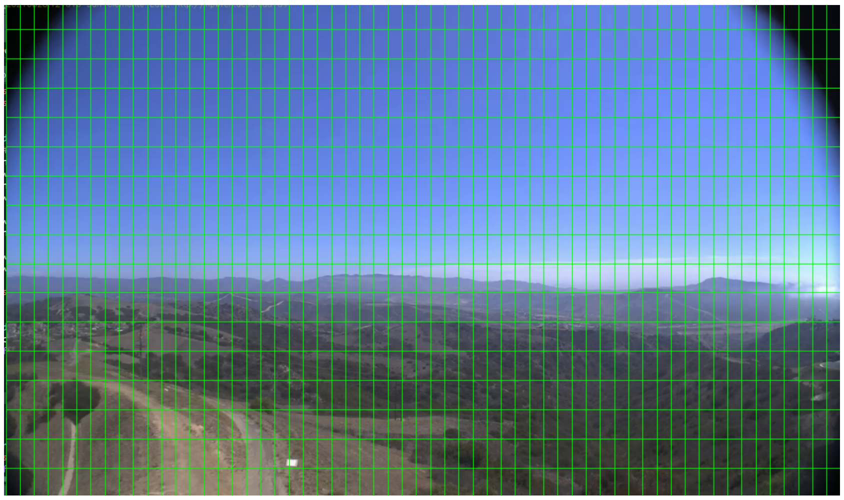

Figure 4 **| Meshgrid**

Following this, we need to establish the regions in which the movement occurs. This is necessary, as using only parts of the frame where there is movement may lead to inaccurate results. Therefore, we use an approach similar to how moving regions are determined. To do this, we divide the frame into regions twice the size of smaller regions, as shown in Figure 5. Next, for each larger region, we calculate the number of smaller regions in which movement was noticed, and if the area of moving pixels exceeds 25% of the larger area, we consider the larger area instead. The areas where movement is determined are then given to the model for *fire* or *non-fire* prediction.

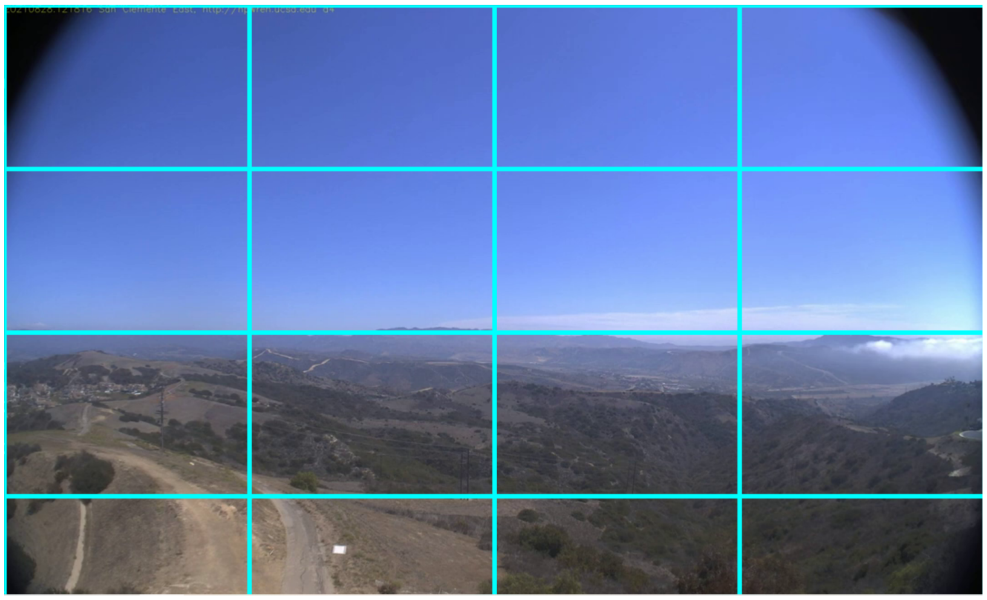

Figure 5 **| Meshgrid of Larger Regions**

**3.2 - Threshold and Background Updating**

As previously mentioned, we analyze each frame and use the motion detection algorithm to find regions where fire or smoke is potentially present. The algorithm consists of two main parts, (1) updating the thresholds and the background, and (2) comparing the current frame with the previous one (*background subtraction*). To update the average background and thresholds, we follow the proposed algorithm of Toreyin, B. U. [1]

First, we initialize the Threshold Image (T), with the same size as the input frame from the video and predetermined value. Next, we create an additional variable called the Background Image (B), and assign the first frame of the video to it as well. Following on to the next frame, we start to determine if the pixel is stationary (no change in the value of the pixel) or not, as calculated by Equation 1, where $I$ is the frame from the video, $x$ is the position of the pixel and $n$ is the nth video frame.

$$(1)\quad |I(x, n) - B(x, n)| > T(x, n)$$

A pixel is considered to be stationary if the absolute difference between the value of the pixel and the value of the same pixel from the previous frame is less than the value from the given Threshold Image at

the same position as the pixel. Instead, a pixel is considered to be moving if the difference is bigger than the value of the Threshold Image.
Next, we update the Background Image (B) and the Threshold Image (I) as given by Equations 2 and 3, where $a$ is a predetermined constant that is positive and close to 1.

$$(2)\ \ Background(x,\ n\ +\ 1)\ \rightarrow\ a\ \cdot\ B(x,\ n)\ +\ (1\ -\ a)\ \cdot\ I(x,\ n)\ ,\ \textit{for a stationary pixel}$$
$$(3)\ \ Background(x,\ n\ +\ 1)\ \rightarrow\ B(x,\ n)\ ,\ \textit{for a moving pixel}$$

Similarly, we update the Threshold Image (I), asn given by Equations 4 and 5, where $a$ is a positive predetermined constant close to 1, and $c$ is a positive real number larger than 1.

$$(4)\ \ Threshold(x,\ n\ +\ 1)\ \rightarrow\ a\ \cdot\ T(x,\ n)\ +\ (1\ -\ a)\ (c\ |I(x,\ n)\ -\ B(x,\ n)|)\ ,\ \textit{for a stationary pixel}$$
$$(5)\ \ Threshold(x,\ n\ +\ 1)\ \rightarrow\ T(x,\ n)\ ,\ \textit{for a moving pixel}$$

Finally, we determine if a pixel is considered to be moving by subtracting the updated Background Image (B) from the current image, for each pixel, thus making a final decision, as calculated by Equation 1.

## 3.3 - Determining Regions with Movement

After determining which pixels are moving, we must determine if the number of *moving* pixels is enough to conclude that there is movement and not simply a result of some distortion on the video. To do so, we divide the frame into segments (squares) whose sides will be 6.6% of the width and length of the frame. The subsequent analysis finds even bigger segments in which the area of moving segments is greater than the 20% of the area of the bigger segment. The size of the segments is double the size of small squares. Both smaller and bigger segments are sent to the model to determine the probability that the fire is present in the segment. An example of *moving* pixels is highlighted in Figure 6.

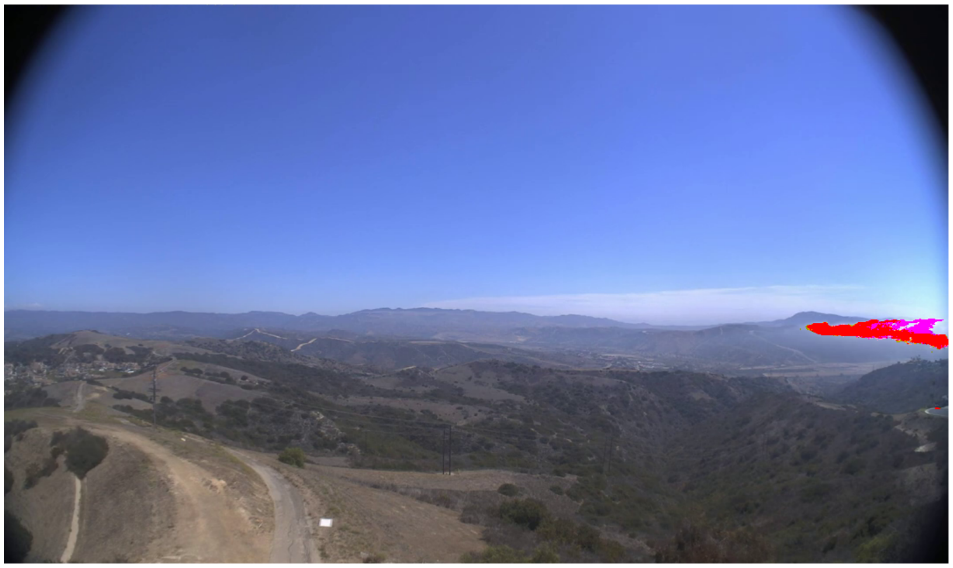

Figure 6 **| Example of *Moving* Pixels Highlighted in Shades of Red**

### 3.4 - Determining *Fire* or *Non-Fire* using a Model

In order to determine whether a specific frame of a video contains fire or not, we decided to extract relevant features that may allow a model to differentiate between a *fire* and a *non-fire* image. In this case, and as evidenced by Figure 7, we concluded that the histograms for the RGB and HSV channels of a *fire* and a *non-fire* image differ significantly so as to be used as representations of the images.

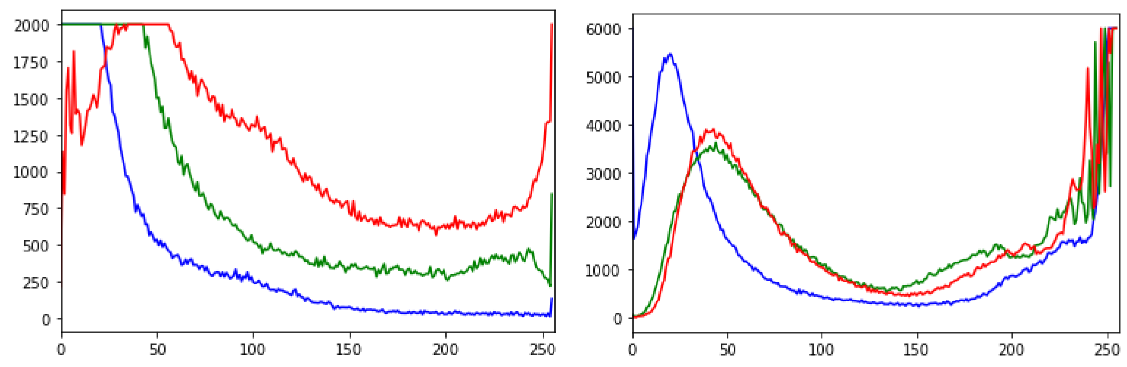

Figure 7 **| Histogram of (a) a Fire Image vs. (b) a Non-Fire Image**

We decided to train a model using labeled image datasets [2] [3] containing *fire* and *non-fire* images. For each image, we calculated the histograms for the R, G, B, H, S, and V channels. As observed in Figure, these histograms follow a *signal* or *wave-like* pattern. To make the model run faster , which is needed for real-time prediction, the histograms were further compressed into relevant features using a Discrete Wavelet Transform (DWT).

The DWT decomposes a signal into corresponding frequency sub-bands, which can then be used to reconstruct the original signal. In our particular case, the signal is simply the histogram of a given channel. Given that we work under the assumption that the histograms differ between *fire* and *non-fire* images, the frequency sub-bands will be different as well.

Following an approach of Taspinar A. [4], we extract features from the frequency sub-bands of each histogram by simply calculating a list of statistics, which includes multiple percentiles, median, mean, standard deviation, variance, and root mean square of the sub-bands. This produces a total of 54 features per image, which can then be given to a model.

As explained, a DWT was applied to each image of our datasets, yielding an overall dataset of 4,507 image samples, represented by 54 DWT features each.

The feature dataset was used to train an XGBoost model. XGBoost stands for *eXtreme Gradient Boosting*, and is a decision tree, ensemble-based model that employs boosting to combine multiple *weak* decision trees to create a *strong* model [5].

To train the model, the dataset was split into a Train Set (60% of data), a Validation Set (20% of data), and a Testing Set (20% of data). First, we performed hyperparameter tuning of the model by training on the Train Set and calculating error using the Validation Set. Using a Bayesian Optimization approach, the relevant parameters of the model were tuned [6]. The model was then tested using the Testing Set.

The fully trained XGBoost model was incorporated into the motion detection algorithm, and each *moving* rectangle was then given to the model to predict whether the given region contained *fire* or not.

## 4 :: Results

After performing hyperparameter tuning on the XGBoost model, a final model was trained using the best parameters as determined by Bayesian optimization. The model was evaluated using the testing dataset, and an accuracy score was calculated. For this, the prediction scores given by the model were converted to categories. We consider a threshold of 80%, where any score above this threshold is considered *fire*, and any score below is considered *non-fire*.

The calculated accuracy score of the model was 93.13%, rounded to two significant figures. To further explain the model's performance, a confusion matrix was generated, which can be observed in Figure 8.

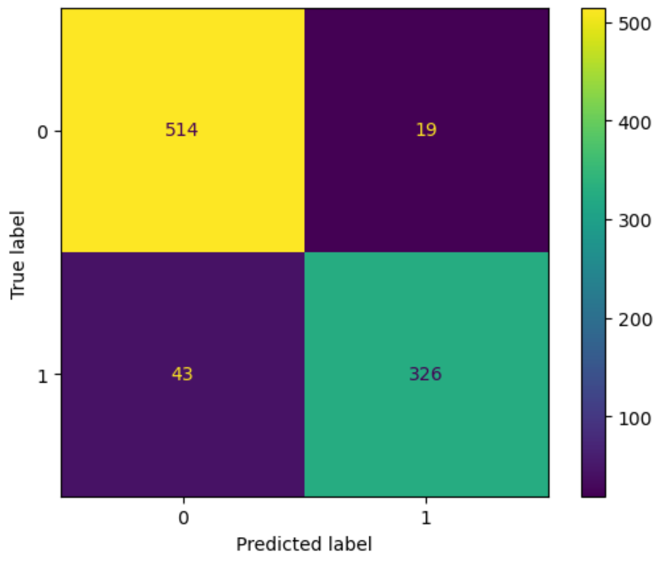

Figure 8 **| Confusion Matrix for Model Testing**

Figure 9 shows an example of the model predicting in real-time in a given video. The highlighted squares are determined by the motion detection algorithm. These are then given to the model, which produces a prediction score shown as well.

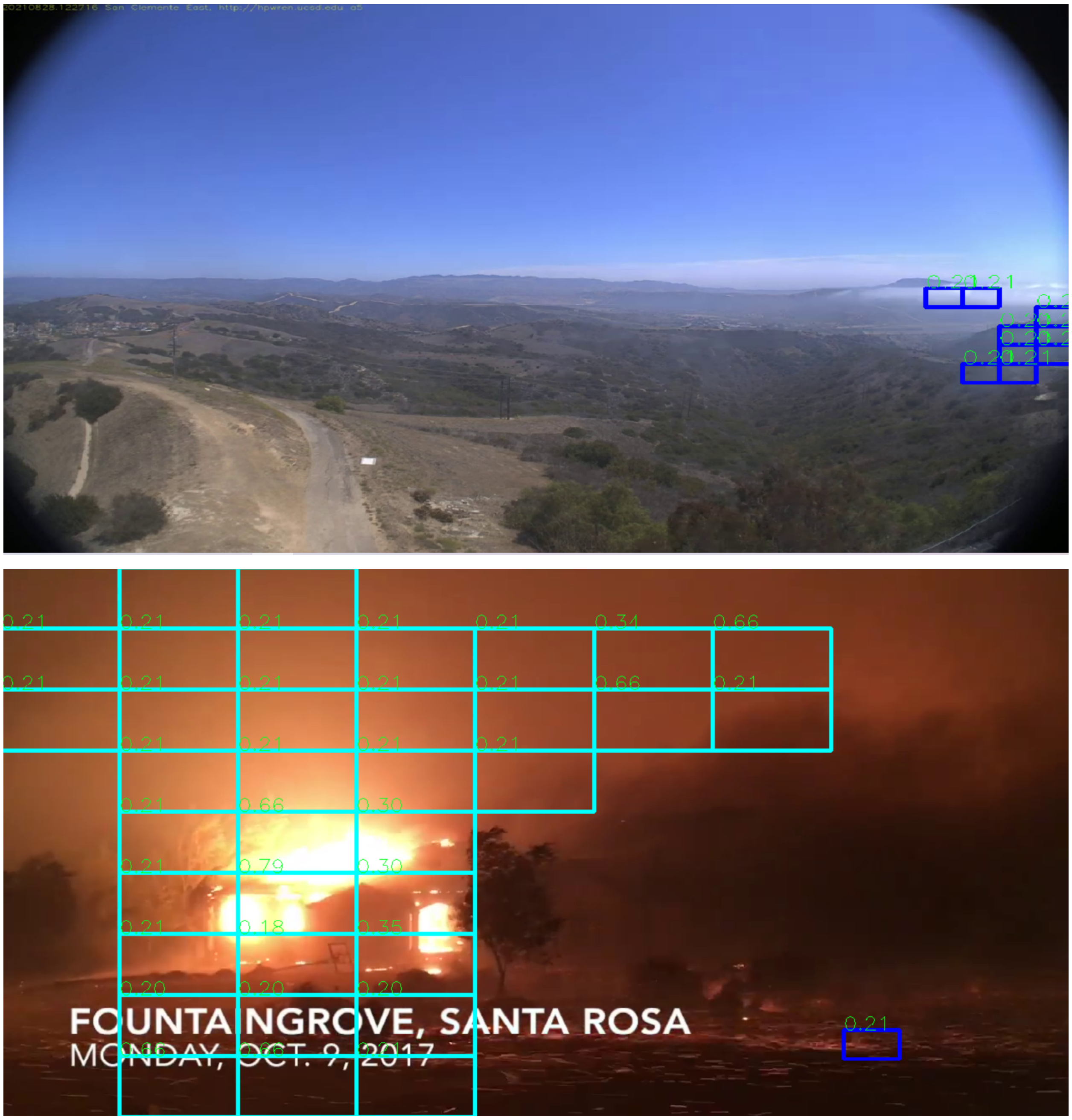

Figure 9 **| Model Prediction**

An example of how the model works on areas with potential movement as determined by the motion detection algorithm. The chance that there is a fire in the given region is predicted by the model and shown above each square.

## 5 :: Conclusions

Based on the explained results, we can conclude that the use of histograms as representatives for *fire* and *non-fire* images is an effective strategy that allows models, like XGBoost, to differentiate between both classes and perform accurate binary classification. Furthermore, this is complemented by a motion detection algorithm for real-time predictions on video cameras, which can be useful for automated forest fire detection.

---

## 6 :: References

[1] Toreyin, B. U. (2009). *Fire Detection Algorithms Using Multimodal Signal and Image Analysis* (dissertation). Retrieved from: https://theses.eurasip.org/theses/224/fire-detection-algorithms-using-multimodal-signal/

[2] Kutlu, K. (2021, March 15). *Forest fire*. Kaggle. Retrieved from: https://www.kaggle.com/datasets/kutaykutlu/forest-fire

[3] Ever, Enver; Yatbaz, Hakan Yekta ; Kizilkaya, Burak; Yazici, Adnan (2020), *Effective Forest Fire Detection Data-set for Heterogeneous Wireless Multimedia Sensor Networks*, Mendeley Data, V2, doi: 10.17632/g5nzp6j3bt.2

[4] Admin. (2021, August 16). *A guide for using The wavelet transform in machine learning*. ML Fundamentals. Retrieved from: https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[5] XGBoost. (n.d.). https://xgboost.ai/

[6] Nogueira, F. (2014). *Bayesian Optimization: Open source constrained global optimization tool for Python*. Retrieved from: https://github.com/bayesian-optimization/BayesianOptimization