Fire Detection Based on AESFERM Framework

Salman Faiz Hidayat[1], Bambang Abhinawa Pinakasakti[2],
Salwaa Mumtaazah Darmanastri[3], Rama Andhika Pratama[4]

[1, 2, 3, 4]Department of Computer Science, Faculty of Mathematics and Natural Sciences,
Gadjah Mada University, Yogyakarta, Indonesia
Email: salmanfaizhidayat@mail.ugm.ac.id [1], bambangabhinawapinakasakti@mail.ugm.ac.id [2],
salwaamumtaazahdarmanastri@mail.ugm.ac.id [3], ramaandhikapratama@mail.ugm.ac.id [4]

NIM: 23/511544/PA/21817 [1], 23/511433/PA/21794 [2], 23/516172/PA/22060 [3], 23/517333/PA/22175 [4]

***Abstract*—Forest fires have been a critical issue globally which cause destructive environmental, economic, and health impacts. Most of the conventional fire detection systems, including smoke and temperature sensors, are slow, expensive, and prone to inaccuracies. This report presents a novel fire detection system that combines the power of Artificial Intelligence and Computer Vision, drawing on features from video data captured by CCTV cameras. The methodology includes image enhancement comprising Gaussian filtering and sharpening, segmentation using Otsu's adaptive thresholding, and feature extraction using SIFT, LBP, and Gabor filters. Temporal changes are analyzed using optical flow, while classification is achieved using Extreme Gradient Boosting. The system tested on the VisiFire dataset showed an overall accuracy of 90%, with the highest performance in detecting forest fire, smoke, and controlled fire. However, limitations were observed for some classes that were underrepresented in the dataset. The performance demonstrated the reliability and scalability of the system for real-time applications. Its promising results, however, need further preprocessing steps and computational efficiency checks for practical real-world applications.**

***Index Terms*—forest fire detection, AI and CV, Gaussian filtering, adaptive thresholding, SIFT, LBP, optical flow, XGBoost, Gabor filter, HSV, F1 score.**

I. Introduction

Forest fires have been a critical global issue with destructive effects on the environment, economy, and human health. Forest fires have burned twice as much tree cover today than they did two decades ago. According to data from researchers at the University of Maryland, the areas that have been burned by forest fires increased by 5.4% per year from 2001 until 2019 [23]. In total, we have lost around 6 million hectares of tree cover per year than we did in 2001.

Various of conventional methods have been implemented to prevent forest fires such as smoke and temperature detectors. However, these methods are not only high in cost, but also slow on detecting forest fires. Additionally, these methods often produce false alarms. Therefore we need a more reliable and accurate method to prevent forest fires from happening.

With the swift evolution of Artificial Intelligence (AI) and Computer Vision over the past years, researchers have implemented the aforementioned technologies to Closed Circuit Television (CCTV) cameras in order to detect various objects, including fire. Such system can detect specific objects by analyzing it's features such as color, motion, and shape. After analyzing the features, the machine will determine the object based on its features that have been observed. Therefore, we can use such technologies and implement it to CCTV cameras that overlook the forest to detect forest fire.

AI and CV have significant advantages over the conventional smoke and temperature sensors in forest fire detection. Conventional systems often suffer from limitations such as delayed response times, high rates of false alarms, and an inability to provide detailed spatial information. On the other hand, AI and CV systems use advanced algorithms to analyze visual data from CCTV cameras in real time, therefore allowing much quicker and more effective smoke, flame, or heat signature detection. The result is not only a quicker response but also we will able to find the location of the fire outbreaks, which is quite crucial to prevent forest fires from happening. [27] [3]

Detection using AI and CV for detecting forest fires are lower on cost rather than conventional methods. Using CCTV cameras that have been implemented with fire detection algorithms are much more scalable and flexible because it can cover a large amount and possibly remote areas to detect forest fire. Conventional methods such as smoke and temperature sensors are limited in range and accuracy. Therefore, making it harder to cover large areas because we would need a large amount of them, increasing the cost while modern methods can cover large areas easily. [11] [26]

AI-driven forest fire detection often implements handcrafted features like motion and color with machine learning algorithms, such as support vector machines and random forest classifiers, for fire detection. One of the key works proposes a real-time fire detection framework that combines color probabilities, motion analysis, and shape recognition. This approach uses machine learning to classify and verify fire occurrences, achieving a true-positive rate of 89.97% and a false-negative rate of just 10.03%. The system was designed to operate quickly, processing an average of 21.7 frames per second, making it highly suitable for real-time monitoring. [26]

Our method, however, uses Extreme Gradient Boosting

(XGBoost). Studies shown that XGBoost is best to use in object detection due to its high accuracy. A researcher once compared XGBoost and random forest in urban land classification using remote sensing data. They found out that XGboost outperformed random forest with an accuracy of 81% while the latter method had an accuracy of 77% [22].

II. Theoretical Basis and Literature Review

In fire detection systems, gathering visual data is essential since it serves as the foundation for further processing and analysis. The VisiFire dataset, created especially for fire detection research, is used in this project. The dataset includes a variety of events, such as forest fires, controlled fires, and non-fire scenarios like red objects and sunshine reflections. It is a great dataset for training models to successfully distinguish real fires from false alarms due to varied settings [1].

After visual data is obtained, it frequently has to be improved to fix problems like noise, blurriness, or bad lighting that can mask important aspects of pictures. Improving image quality makes it easier to identify important details for additional research, such as smoke patterns or flame edges. This project uses sharpening techniques to enhance edge sharpness and Gaussian filtering to reduce noise [9]. In challenging scenarios involving erratic lighting or motion artifacts, augmentation has the potential to improve detection accuracy [6]. The image is smoothed by Gaussian filters, which reduce random noise while maintaining crucial structural information. Similarly, sharpening methods intensify edges, making important patterns like flame boundaries more visible.

To complement the enhancement step, Otsu's adaptive thresholding method is utilized for segmentation to address variations in lighting and pixel intensity. By dynamically adjusting thresholds based on local pixel distributions, the method ensures sensitivity to both bright and dim areas. This technique flexibly handles differences in image quality and illumination, accurately isolating flame or smoke regions while excluding irrelevant background features [19] [14]. We can see in Fig. 2, after performing the otsu's thresholding, it is possible to obtain the potential forest fire point monitoring threshold for each sub-region.

To recognize the characteristic of fire in a given frame, a method that will be used is blob detection. The reason for this selection is the fact that it can capture points with varying brightness and color as shown in Fig. 3. The fire in the data set is recorded outdoors with the possibility of varying illuminations, and blob detection can exactly solve this. Furthermore, it can be combined with adaptive thresholding, such as the selected Otsu method mentioned here. And finally, blob detection can dynamically compute a threshold image based on local properties [17] [18]. This fits our problem because the fire is irregular in shape. Once the blob detection recognizes the fire features, the SIFT algorithm will be performed to extract the key features of the detected blobs. This will allow for easier object detection, or fire in this case. The algorithm involves four main steps. The first one is Scale-space extrema detection, which detects potential points at different

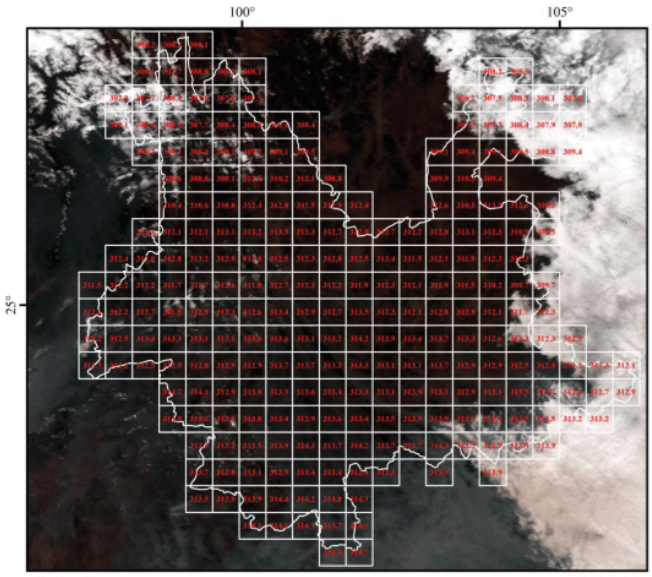


Fig. 1: Example of otsu thresholding in fire detection [10]

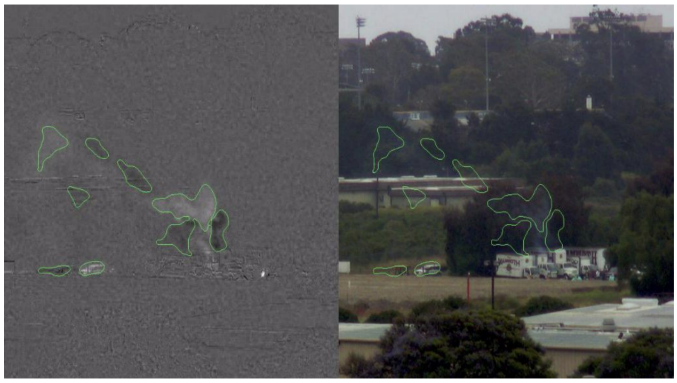


Fig. 2: Example of blob detection in fire detection [17]

scales using the Difference of Gaussians (DoG). The second step is keypoint localization, where keypoints are refined by eliminating low-contrast or poorly localized candidates. The third step is orientation assignment, which assigns a consistent orientation to each keypoint to achieve rotation invariance. The final step is keypoint descriptor generation, where a unique descriptor vector is computed for each keypoint, enabling reliable matching across different images. After the SIFT algorithm captures all the points of interest (POI) within the blobs, it will then be compared with another image for feature matching. Feature matching is useful for detecting movements and recognizing objects between two images [15] [2]. Example of which is shown in fig. 4. After the features have been extracted, they must be represented well. For colors, the color space that is used to represent the colors in the frame is HSV (Hue, Saturation, and Value). This color space was chosen because it closely resembles to how humans perceive colors [26]. This particular color space should be able to capture colors that are closely related to fire, such as red, orange, and

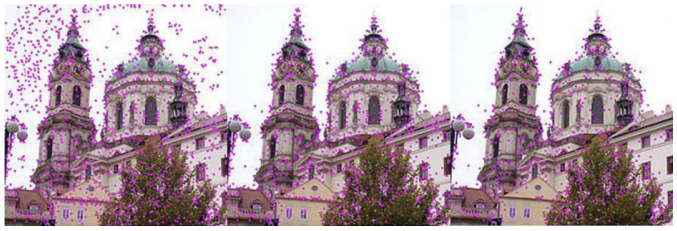
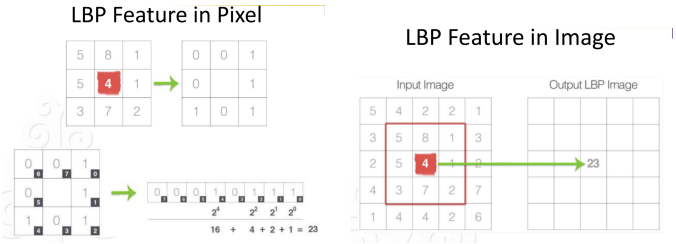


Fig. 3: Example of keypoints detected by SIFT algorithm. [15]



(a) Binary arrangement and converting to decimal.

(b) Replacing new image with the converted decimal

Fig. 5: How LBP works in general [24]

yellow [16]. For texture, it is represented using Local Binary

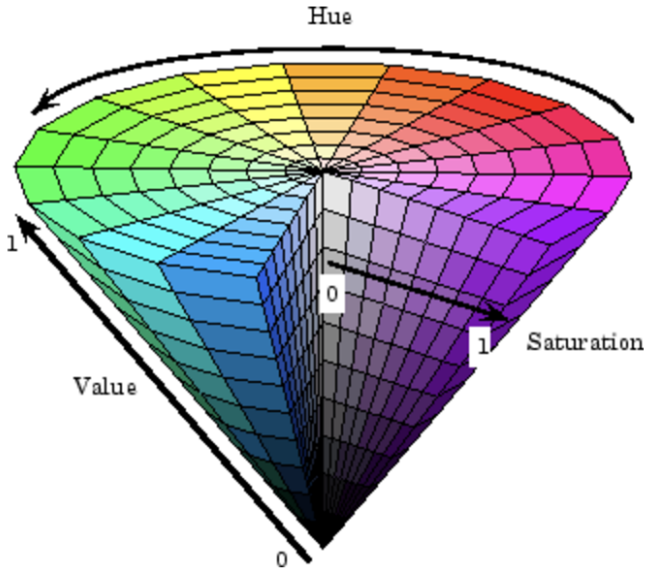


Fig. 4: Illustration HSV color space [16].

Pattern (LBP). It works by thresholding the neighboring 3x3 pixel value. If it is greater than the pixel in the center, it becomes 1, otherwise 0. Then, the 8 surrounding binary values will be rearranged and converted to decimal. The resulting decimal will replace the pixel value in the center, for the new image. However, for the proposed method, the LBP image will not be used directly. Rather, the histogram will be taken as the feature for the subsequent model that will train on it [24] [25]. The working of LBP and histogram extraction are shown in fig. 5 and 6. Alongside LBP, Gabor filter is used, which is defined by the equation in fig. 7. The reason is due to Gabor filter's flexibility as a kernel. Various types of kernel could be generated from one function, by only changing the parameters. It detects edges and textures in the image [4]. An example of its application is shown in fig. 8.

Lastly, to represent the dynamics of fire, a feature representation method named optical flow will be used. Optical flow works by calculating a vector for each pixel, which represents the direction and magnitude of the image [13] [12] [5] as shown in fig. 9.

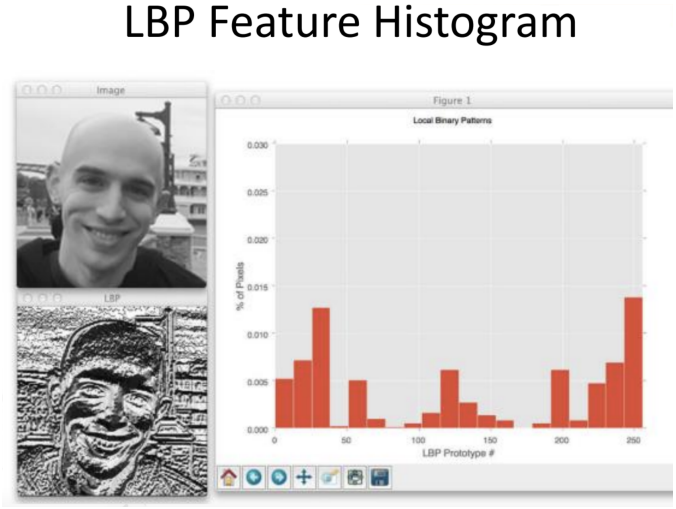


Fig. 6: Taking the histogram of LBP. [24].

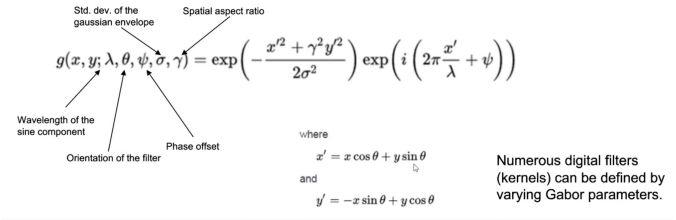


Fig. 7: The equation that Gabor filter follows. [4].

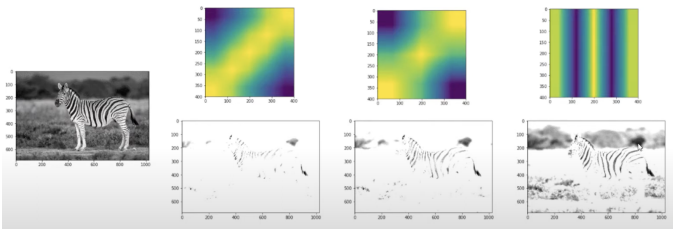


Fig. 8: Examples of kernels produced from Gabor filter. As an example, the first kernel (the left-most kernel) has the θ parameter set to $45°$, while the third kernel has θ set to $90°$ [4].

Fig. 9: Live calculation of vectors for each pixel in optical flow. [13].

III. Methodology

The proposed fire detection system uses an integrated pipeline based on digital image processing techniques to detect fire in video frames. The process systematically uses each frame to enhance features, outline regions of interest, and extract relevant features via static and dynamic analyses. The significant steps are addressed as follows:

A. Data Preparation

The dataset that was used is from the VisiFire dataset from Bilkent University [1]. It consists of folders with videos of fire and smoke. The dataset includes fire clips, smoke clips, forest smoke clips, and additional clips if needed.

Before the data are processed, they must first be labeled. The labeling is done based on the subfolders, which represents a specific class that the model has to predict. Note that these subfolders exist within a parent folder named ”visifire”:

- AccidentsInATunnel: A video of an accident in a tunnel which produces smoke (label: 0).
- FireClips: Various video clips of fire from small to large fires (label: 1).
- OtherClips: Additional clips of car counting and barbecue video(label: 2).
- Smoke_Far: Video clips of smoke that is recorded from far away (label: 3).
- SmokeClips: A collection of smoke clips in general (label: 4)

Afterward, a function named acquire_videos_and_labels will traverse each subfolder and give each path a label based on the folder name. Here is an example:

- File path: [”/path/FireClips/video1.mp4”, ”/path/Smoke_Far/video2.mp4”]
- Labeling: [1, 3]

Following that, the dataset is split into train and test (80% training sample, 20% testing sample). And finally, each frame of every video is processed through the proposed method.

B. Image Acquisition

Video frames are extracted sequentially using OpenCV’s *VideoCapture* method. This ensures a structured dataset of continuous frame for temporal analysis.

C. Image Enhancement

To improve the visibility and distinguishability of fire features, the frames are processed as follows:

1) Gaussian Filtering: Removes high-frequency noise while retaining edge information.
2) Image Sharpening: Enhances the contrast of edges using a convolutional kernel to make fire patterns more noticable.

D. Image Segmentation and Convex Hull Analysis

Segmentation uses adaptive thresholding via Otsu’s method, effectively isolating fire-like regions from the background. Subsequently, compute convex hulls for the segmented contours to encapsulate irregular shapes, ensuring better identification of fire’s dynamic, amorphous structure.

E. Blob Detection

Using the Difference of Gaussian (DoG) method, Blob detection identifies regions with varying brightness and size. This approach is particularly suited to capturing the irregular, flame-like regions characteristic of fire.

F. Feature Extraction

The extracted features include:

1) Scale-Invariant Feature Transform (SIFT): Detects keypoints and computes descriptors for high-contrast areas, capturing the fine-grained details of flames.
2) Local Binary Pattern (LBP): Characterizes texture by encoding patterns of local pixel intensity changes into histograms.
3) Gabor Filter Responses: Extracts texture features by analyzing images with multi-orientation kernels to detect edges and textures associated with fire.

G. Feature Representation

The visual features are combined to form a comprehensive representation:

- Color Features: Fire-relevant hues such as red, yellow, and orange are captured using the HSV color space.
- Texture Features: LBP histograms provide detailed texture analysis, complementing the color information.
- Dynamic Features: Temporal changes are captured using the dense optical flow method (Farneback), where histograms of flow magnitudes represent the movement patterns in the scene, essential for identifying fire’s dynamic behavior.

H. Testing on Diverse Fire Scenarios

The method is tested on video from the following scenarios for visual analysis:

1) Controlled Fire: This method includes outlining designated fire zones in a controlled environment. These tests measure the technique's ability to detect features of fire where the background is more or less straightforward.
2) Forest Fire: This class includes videos of wildfires with large-scale flames and smoke within natural elements such as trees. In these cases, system efficiency is tested to detect fire effectively within a cluttered environment.
3) Barbecue Fire: small-scale fire with complicated geometries and unsteady dynamics. This test proves the approach's capability to capture the localized and fine details of fire.
4) Backyard Fire includes fires in a semicontrolled outdoor environment with various illuminations and motions.
5) Erratic Flames (Fire.1) are dynamic fires with fluctuating intensities and movements. The method tracks fire regions well and captures boundary details across frames.
6) Smoke Detection: This method detects diffused, transparent smoke cases. It splits smoke regions accurately, although some of the boundary information is not distinct.

These test cases examine the system's robustness to illumination and scene dynamics changes. Each video scenario enabled the pipeline to showcase its ability to isolate and recognize fire regions, regardless of scale, background complexity, or fire dynamics. This extensive testing ensures the generalizability of the system to all kinds of fire detection application.

I. Label-Feature Alignment, Feature Matrices, and Normalized Features

After the features of each frame are obtained, they should now be associated with the label of the video, from which the frame was extracted (the label given at the before the preprocessing steps). Then, the features are converted into feature matrices (X_train and X_test) and label arrays for the target of the prediction (y_train, y_test). Lastly, the features are normalized so that the model can achieve better generalization after training [7].

J. Modeling

The machine learning algorithm that was chosen to make the detection is XGBoost. XGBoost is used because it was shown that it performs well in object detection, including fire [21] [20]. XGBoost is a tree-based model that uses the ensemble method to make predictions. It first starts with one model, making a naive prediction for each observation in the dataset. Then, the loss function is then calculated to fit a new model that minimizes the loss function. Afterwards, the new model will be added to the ensemble, and the ensemble will create a new prediction. This will be repeated for a defined number of rounds, specified by the n_estimator parameter. The following is the parameter that was set prior to training:

- "objective": "multi:softmax",
- "num_class": 5,
- "tree_method": "gpu_hist",
- "predictor": "gpu_predictor",
- "max_depth": 6,
- "learning_rate": 0.1,
- "n_estimators": 100,

Fig. 10 is the flow chart of how XGBoost model works:

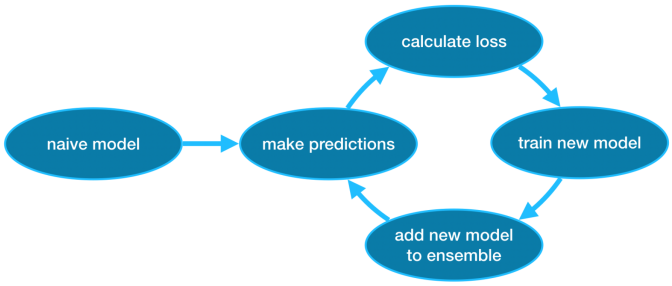


Fig. 10: Flowchart of XGBoost. [8].

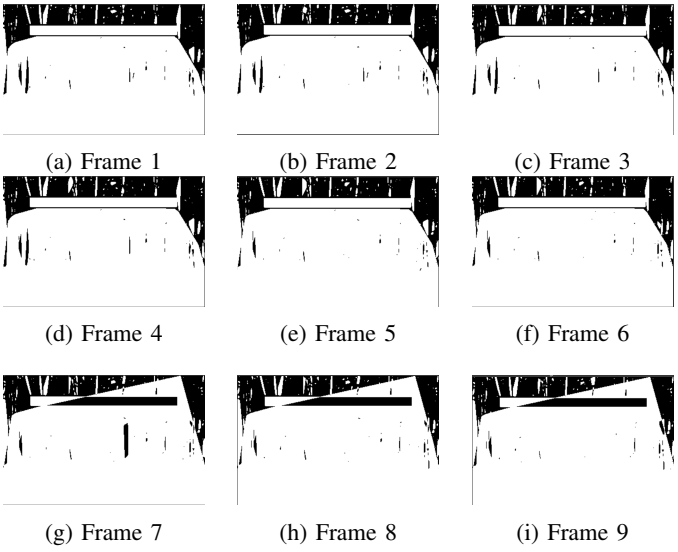
IV. Experimental Result and Discussion

The experiment was conducted by testing the proposed method on the VisiFire dataset [1]. The method was tested with various cases of fire videos: controlled fire, forest fire, small fire (such as barbecue and backyard fire), and various smoke videos. This section will display the sample frame from each case and the output frames from the proposed method.

Firstly, the proposed method was tested with the controlled fire. There are 2 videos of controlled fire, where one fire is bigger in one video than in the other. It seems that the method recognized most of the image as foreground, as the smaller fire did not stand out as much, as shown in Fig. 11 and 12, a similar case also happens to a small barbecue fire in Fig. 15 and 16. However, when the controlled fire is bigger, the proposed method could distinguish the fiery parts of the image more clearly shown in Fig. 13 and 14. However, the proposed

Fig. 11: Example frame from a small controlled fire video in the VisiFire dataset [1].

method resulted in a fairly good fire detection for forest fire.



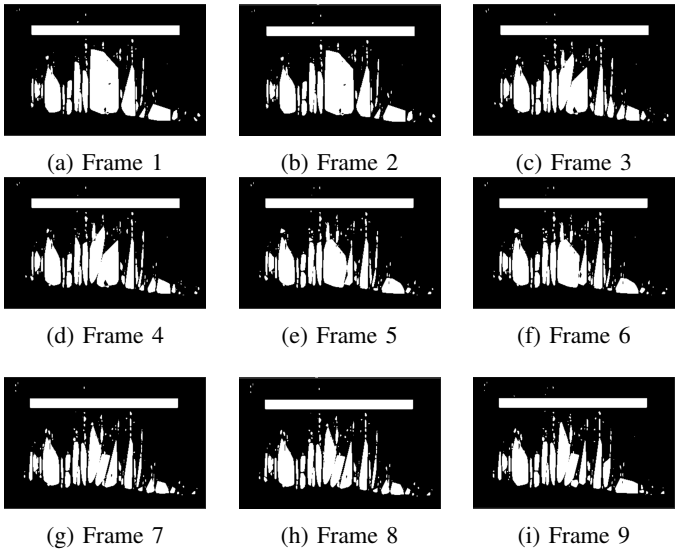
(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 12: Example output from the small controlled fire. Most of the image is considered as foreground, not just the fire.

Fig. 13: Example frame from a big controlled fire video in the VisiFire dataset [1].



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 14: Example output from the big controlled fire. It can be observed that the fire region in the image is detected clearly (considered as foreground in white).

Fig. 15: Example frame from a barbecue fire video in the VisiFire dataset [1].

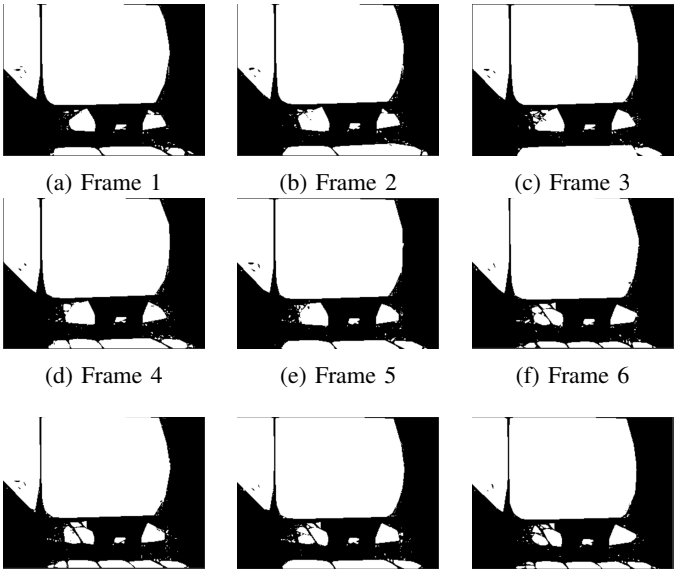
For the case of forest fire, the method highlights the fire section of the image well with its convex hull algorithm. One could hypothesize that the method recognizes the irregular shape of fire and smoke. Another thing to note is that the large smoke in the background behind the trees. The method also clearly ignores the trees in the background, which is a good indicator that it focuses on the fire. The sample frame is shown in fig. 17 and fig. 18 shows 9 frames as a result from the proposed method.

In addition to the controlled fire, barbecue, and forest fire scenarios, the proposed method was further evaluated using additional cases, including a backyard fire, fire.1, and various smoke videos.

In the backyard fire video, the method effectively highlighted the fire regions, although some parts of the background were occasionally misclassified as foreground. Despite this, the detection was generally accurate, with the fiery regions standing out clearly in the frames, as shown in Fig. 19 and fig. 20 shows 9 frames as a result from the proposed method.

For the fire.1 case, the flames exhibited erratic movements and dynamic changes in intensity. The results showed that the method successfully tracked the fire regions across frames. The flame boundaries were well-defined, and the method consistently captured the key features of the fire, even in challenging frames, as illustrated in Fig. 21 and fig. 22 shows 9 frames as a result from the proposed method.

The smoke detection tests presented a more complex challenge due to the diffused and translucent nature of smoke. While testing the ManavgatTEstSmoke case in Fig. 23, our results revealed a significant limitation - instead of detecting the actual smoke in the scene, I think the method incorrectly classified white bridge structures as smoke regions, highlight-

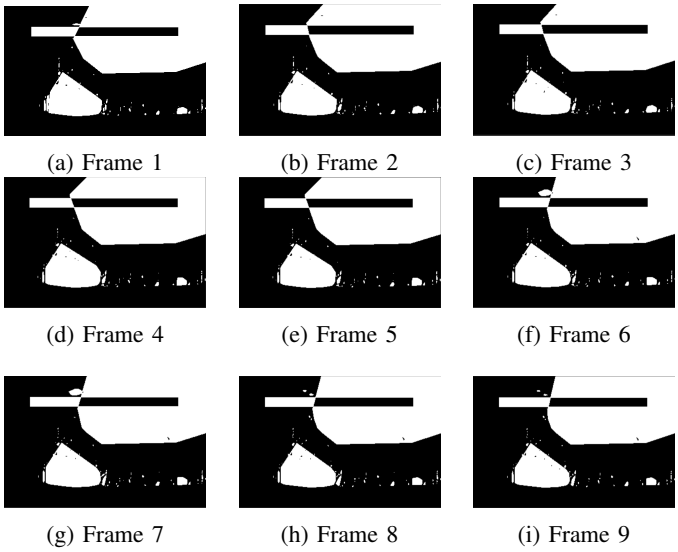


(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 16: Example output from the barbecue fire. It can be observed that the foreground is not necessarily always fire. It detects almost the top half of the image as foreground.



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 18: Example output of forest fire detection pipeline using the proposed method. The region of fire around the bottom-left side of the image can be highlighted clearly, as well as the fumes of smokes around the top right of the image.

Fig. 17: Example frame from a forest fire video in the VisiFire dataset [1].

Fig. 19: Example frame from a backyard fire video in the VisiFire dataset [1].

ing a critical weakness in distinguishing between static white structures and genuine smoke patterns. As shown in Fig. 24, overall, the results indicate that the method failed to detect any actual smoke patterns.

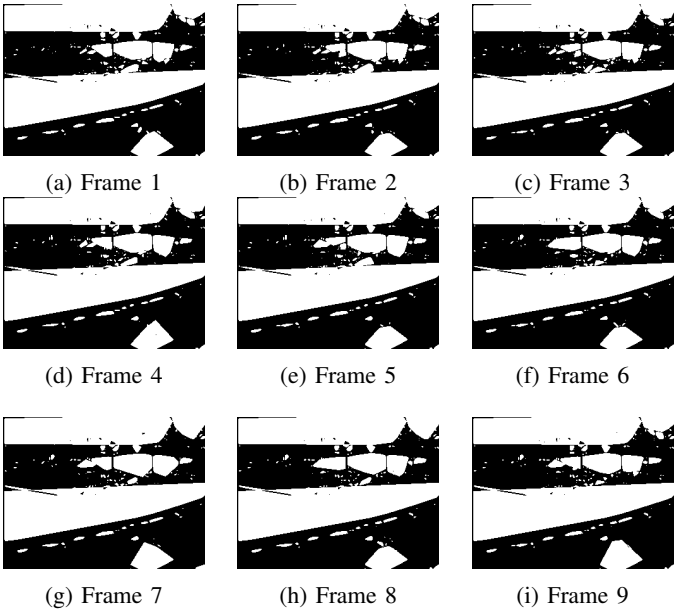
Besides the look, interesting things to note is that the proposed pipeline produced some feature vectors for a model to train on. These include:

- Convex Hulls found
- Detected n blobs
- Extracted SIFT Descriptors
- Optical Flow Histogram Length
- Feature Vector Length

Regarding the model, One stark observation that could be made is that the model had no support vector for category 0 (AccidentInATunnel) and category 2 (OtherClips). It can be hypothesized that since for those 2 categories, there are significantly less frames for the model to train on. Therefore, it could not make any prediction for those 2 categories.

Thus, for the other categories where there are a lot of frames, the model could generalize its prediction well. For category 2 (FireClips), the model achieves an F1 score of 0.74, category 3 with F1 score of 0.95 (Smoke_Far), and category 4 with an F1 score of 0.84 (SmokeClips).

The model adopted in this method is XGBoost. The two

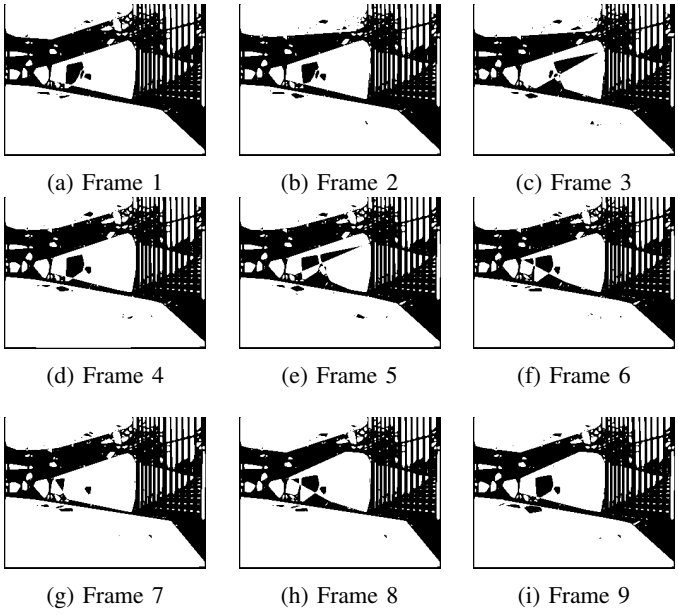


(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 20: Example output from the backyard fire.



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 22: Example output from the fire.1.

Fig. 21: Example frame from a fire.1 video in the VisiFire dataset [1].

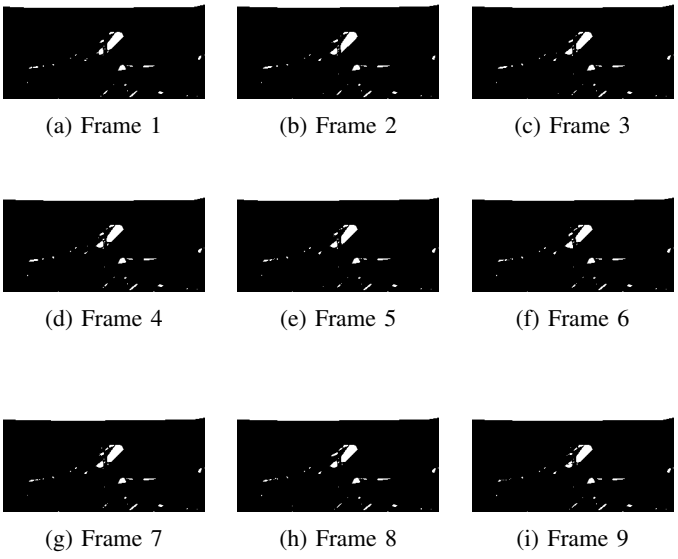
Fig. 23: Example frame from a manavgatTEstSmoke video in the VisiFire dataset [1].

classification reports show how good the proposed method is with and without HSV features, pointing out the strengths and weaknesses of each method. In the first report (Fig. 25), in the method without HSV being shown, the model performs quite well for categories with sufficient training data but could be better where there is little data. In particular, it does not make any predictions for Category 0 (AccidentInATunnel) and Category 2 (OtherClips) since they are not supported. This probably goes down to the few frames available in the training dataset for such categories, giving the model a limited ability to learn and apply its knowledge.

For the other categories, the method without HSV works well. In category 1 (FireClips), the model gets a precision of 0.68, a recall of 0.81, and an F1-score of 0.74, showing moderate success. At the same time, categories 3 (Smoke_Far) and 4 (SmokeClips) have much higher F1-scores of 0.95 and 0.84, respectively. The overall performance is good, with a weighted average F1-score of 0.91 and an accuracy of 90%. However, this approach incurs a heavy computational cost: running the pipeline on a Kaggle notebook with T4 x2 GPUs takes almost two hours to indicate the need for optimization.

The second report is for the proposed method using HSV features. It has similar computer needs, taking about two hours to run with the same resources.

This method depicts slightly different performance results. Adding HSV features drops the overall accuracy to 85% and a weighted average F1-score of 0.85. The performance in each category also changes, with a clear trade-off between



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

(g) Frame 7 (h) Frame 8 (i) Frame 9

Fig. 24: Example output from the manavgatTEstSmoke.

precision, recall, and F1 scores. That change means that while HSV features may contain helpful information, their combination requires further adjustments to improve things. Such comparison shows that the HSV-oblivious method scores better on average accuracy than F1 scores but worsens the sparse categories. By using the features extracted by HSV, one can guarantee much better balance in the performances across the categories; on the other hand, these are prone to computational problems—it would take nearly two hours just to run. Also, one correction in the caption of Fig. 26 should be made because it refers to a method without HSV, while it is labeled with HSV.

Overall, this is considerably good, given that one possible implementation that uses convolutional neural network (CNN) and XGBoost achieves an accuracy score of around 98.53% [20], an 8.53% difference.

However, an improvement that could be made is that the computation time. Running the whole pipeline in a Kaggle notebook using T4 x2 GPU took more or less 2 hours for both cases. This will not be practical if a real-time fire detection should be implemented. Various ways for the next experiment to improve computational time is to the number of features, increase the samples with low accuracy score, and perhaps a simpler algorithm in either preprocessing or modeling. Furthermore, the preprocessing steps involves a lot of functions with various parameters. Therefore, an improvement that could be made is that parameters should be better adjusted.

V. Conclusion

In this study, various methods of feature extraction and image processing techniques were explored. The preprocessing steps starts from enhancement using Gaussian and sharpening

Classification Report:

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	0
1.0	0.68	0.81	0.74	906
2.0	0.00	0.00	0.00	0
3.0	0.99	0.91	0.95	7051
4.0	0.81	0.88	0.84	2561
accuracy			0.90	10518
macro avg	0.50	0.52	0.51	10518
weighted avg	0.92	0.90	0.91	10518

Fig. 25: Classification report from the proposed method without one of the features (HSV).

Classification Report:

	precision	recall	f1-score	support
1	0.36	0.22	0.27	906
3	1.00	0.91	0.95	7051
4	0.66	0.90	0.76	2561
accuracy			0.85	10518
macro avg	0.67	0.68	0.66	10518
weighted avg	0.86	0.85	0.85	10518

Fig. 26: Classification report from the proposed method.

filter. Next, the image was segmented using Otsu segmentation method combined with convex hulls. Afterwards, a blob detection algorithm was ran to capture the irregular flame-like regions characteristic of fire. Then, features extracted include SIFT (Scale-Invariant Feature Transform), Local Binary Pattern (LBP), and using Gabor filter. SIFT is used to extract key features of the detected blobs, LBP for texture representation, and Gabor filter for better detection of edges and textures. Following that, the fire's dynamic features are obtained using optical flow. And finally, the model XGBoost performs considerably well for a machine learning algorithm.

In conclusion, the proposed method performs well, though some improvements are needed. Those improvements include, simpler feature extraction (as shown by the comparison), simpler and more robust model for prediction, more data for categories that perform poorly, and better adjustments in the preprocessing steps.

References

[1] Computer vision based fire detection software.

[2] Vineeth Balasubramanian. (367) feature matching - youtube, 9 2020.

[3] Sarang Bathalapalli, P. Krishna Prasad, and Ramesh Ponnala. A deep learning approach to forest fire detection and monitoring. *2023 International Conference on Advances in Computation, Communication and Information Technology, ICAICCIT 2023*, pages 263–268, 2023.

[4] Srenivas Bhattiprolu. (367) tutorial 74 - what are gabor filters and how to use them to generate features for machine learning? - youtube, 9 2020.

[5] S. Alam C. Fatichah and D. Navastara. Optical flow feature based for fire detection on video data. In *Proceedings of the 2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pages 100–105, 2019.

[6] T. Celik. Fast and efficient method for fire detection using image processing. *ETRI Journal*, 32(6):881–890, 2010.

[7] Xinyi Chen, Xiang Liu, Yuke Wu, Zhenglei Wang, and Shuo Hong Wang. Research related to the diagnosis of prostate cancer based on machine learning medical images: A review. *International Journal of Medical Informatics*, 181:105279, 2024.

[8] Alexis Cook. Xgboost.

[9] F. Sthevanie D. Lazzaro and K. N. Ramadhani. Enhancing fire detection in images using faster r-cnn with gaussian filtering and contrast adjustment. In *Proceedings of 2023*, volume 7, pages 1572–1581, 2023.

[10] Zhao Deng and Gui Zhang. An improved forest fire monitoring algorithm with three-dimensional otsu. *IEEE Access*, 9:118367–118378, 2021.

[11] Ismail El-Madafri, Marta Peña, and Noelia Olmedo-Torre. Real-time forest fire detection with lightweight cnn using hierarchical multi-task knowledge distillation. *Fire 2024, Vol. 7, Page 392*, 7:392, 10 2024.

[12] S. Fazekas and D. Chetverikov. Analysis and performance evaluation of optical flow features for dynamic texture recognition. *Signal Processing: Image Communication*, 22(7):680–691, 2007.

[13] A. French. Optical flow. YouTube, Oct. 2019.

[14] J. Luo G. Zhang, B. Li and L. He. A self-adaptive wildfire detection algorithm with two-dimensional otsu optimization. *Mathematical Problems in Engineering*, 2020:3735262, 2020.

[15] R. Jin and J. Kim. Tracking feature extraction techniques with improved sift for video identification. *Multimedia Tools and Applications*, May 2015.

[16] MATLAB & Simulink. *Understanding color spaces and color space conversion*, 2024. Accessed: Sept. 26, 2024.

[17] G. F. Moussa. Early forest fire detection using texture, sift threshold, and motion analysis of principal components. Master's thesis, California Polytechnic State University, 2012.

[18] Shree Nayar. (367) detecting blobs — sift detector - youtube, 3 2021.

[19] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[20] U. R, S. S, R. Ganesan, and U. S. Forest fire detection using cnn-rf and cnn-xgboost machine learning algorithms. *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pages 547–553, 2023.

[21] Saydirasulov Norkobil Saydirasulovich, A. Abdusalomov, Muhammad Kafeel Jamil, R. Nasimov, D. Kozhamzharova, and Y. Cho. A yolov6-based improved fire detection approach for smart city environments. *Sensors (Basel, Switzerland)*, 23, 2023.

[22] Zhenfeng Shao, Muhammad Nasar Ahmad, and Akib Javed. Comparison of random forest and xgboost classifiers using integrated optical and sar features for mapping urban impervious surface. *Remote Sensing 2024, Vol. 16, Page 665*, 16:665, 2 2024.

[23] Alexandra Tyukavina, Peter Potapov, Matthew C. Hansen, Amy H. Pickens, Stephen V. Stehman, Svetlana Turubanova, Diana Parker, Viviana Zalles, André Lima, Indrani Kommareddy, Xiao Peng Song, Lei Wang, and Nancy Harris. Global trends of forest loss due to fire from 2001 to 2019. *Frontiers in Remote Sensing*, 3:825190, 3 2022.

[24] Wahyono. Local binary pattern. YouTube, 2024. Accessed: Sept. 26, 2024.

[25] Wahyono. Textured based feature representation, 2024. Digital Image Processing - 10th Lecture.

[26] Wahyono, Agus Harjoko, Andi Dharmawan, Faisal Dharma Adhinata, Gamma Kosala, and Kang Hyun Jo. Real-time forest fire detection framework based on artificial intelligence using color probability model and motion feature analysis. *Fire*, 5, 2 2022.

[27] Berk Özel, Muhammad Shahab Alam, and Muhammad Umer Khan. Review of modern forest fire detection techniques: Innovations in image processing and deep learning. *Information 2024, Vol. 15, Page 538*, 15:538, 9 2024.