CSE-4128: Image Processing and Computer Vision Laboratory

Fire Detection using Image Processing

Name	Md. Masudur Rahman Rabby
Roll	1907113
Section	B
Department	CSE
Batch	2k19
Submission Date	01.07.2024

Objectives:

The primary objectives of this project are:

1. Developing a fire detection application using basic image processing techniques.
2. Creating a user-friendly graphical user interface (GUI) for loading images and displaying results.
3. Detecting fire regions in an image without using machine learning or deep learning methods.
4. Displaying intermediate processing steps and the final detection results clearly.
5. Providing a reliable and efficient tool for automatic fire detection in images.

Introduction:

Fire detection is crucial for safety, surveillance, and emergency response, as early detection can prevent property damage, save lives, and minimize environmental impact. Traditional methods, relying on manual observation or smoke detectors, are often slow and prone to errors. Advances in computer vision and image processing now offer faster and more reliable automated fire detection.

This project develops a fire detection application using basic image processing techniques to identify fire regions in images. Unlike machine learning approaches that require large datasets and significant resources, this application employs straightforward, computationally efficient methods without the need for extensive training data.

The application features a user-friendly graphical user interface (GUI) built with Tkinter, allowing users to load images and process them to detect potential fire regions. The detection process involves converting the image to the HSV color space, applying color-based thresholding, performing morphological operations to reduce noise, and detecting contours to identify fire boundaries.

Each processing step's results are displayed, providing a clear understanding of the fire detection process. The final result indicates whether fire was detected, with visual markers highlighting fire regions. This transparency allows users to adjust processing parameters for different scenarios.

Focusing on fundamental image processing techniques, this project provides an efficient fire detection tool suitable for various settings, especially where quick and reliable detection is essential and computational resources are limited. This project lays the groundwork for further development in automated fire detection systems, potentially integrating more advanced techniques in future iterations.

Functionalities

The fire detection application includes the following functionalities:

- **Load Image:** Users can load an image from their file system.
- **Fire Detection:** The application processes the loaded image to detect fire regions.
- **Display Results:** The application displays intermediate processing steps and the final result, indicating whether fire was detected.
- **Clear Panels:** The application clears previous results when a new image is loaded.

Working Details

The fire detection application employs a series of image processing steps to identify fire regions in an image. These steps are executed within a Python script, leveraging libraries such as OpenCV for image processing and Tkinter for the graphical user interface. Below, we discuss the key functions and their roles in the processing steps:

1. GUI Design

The graphical user interface (GUI) is designed using Tkinter, providing a user-friendly way to load images and display results. The main components of the GUI include buttons for loading images, labels for displaying results, and frames for showing images at various stages of processing.

2. Image Processing Workflow

The workflow for detecting fire in an image involves several key steps, each performed by specific functions within the code. The steps are as follows:

2.1 Load Image

The `load_image` function allows the user to select and load an image from their file system. This function opens a file dialog, retrieves the selected image path, and initiates the fire detection process.

2.2 Clear Panels

The `clear_panels` function clears previous results from the GUI when a new image is loaded. This ensures that the GUI is reset and ready to display the new results.

2.3 Detect Fire

The `detect_fire` function is the core of the fire detection process, involving several key steps:

Step 1: Convert to HSV

Converts the image from BGR to HSV color space using `cv2.cvtColor`. The HSV color space is better suited for color-based thresholding.

Step 2: Define Color Range

Specifies the lower and upper bounds of the HSV values that represent fire colors. These values can be adjusted based on the specific conditions and appearance of the fire.

Step 3: Thresholding

Applies a threshold to the HSV image using `cv2.inRange` to create a binary mask where fire-like regions are white, and other regions are black.

Step 4: Morphological Operations

Uses morphological closing and opening to remove noise from the binary mask. The functions `morphological_close` and `morphological_open` are custom functions that perform these operations using a specified kernel.

Step 5: Find Contours

Detects contours in the binary mask using a custom function `find_contours`.

Step 6: Draw Bounding Boxes

Iterates through the detected contours, calculates the bounding rectangle for each using the custom `get_bounding_rect` function, and draws rectangles around fire-like regions in the original image. This function computes the minimal bounding rectangle coordinates `(x, y, w, h)` for a given contour by finding the minimum and maximum x and y coordinates.

2.5 Display Functions

The display functions show the processed images at various stages on the GUI:

- **display_image**: Displays an image with an optional grayscale conversion.
- **display_contours**: Draws and displays contours on the image.
- **display_output_image**: Shows the final output image with detected fire regions.

These functions create a Tkinter frame for each image, resize and convert the image to a format compatible with Tkinter, and then display it within the frame.

Result and Discussion

The fire detection application successfully detects fire regions in images and highlights them with bounding rectangles. The application displays intermediate processing steps, such as the HSV image, the fire mask, and the contours, providing a clear understanding of the detection process. The outputs are discussed below:

The fire detection application processes and displays several output images at different stages of detection:

1. **Original Image:**

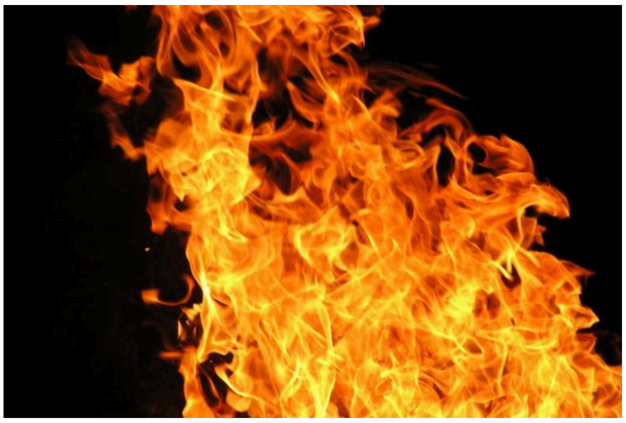

Fig1: Original RGB input image

2. **HSV Image**: Displays the original image converted to the HSV color space. This transformation isolates colors associated with fire, making it easier to identify potential fire regions.

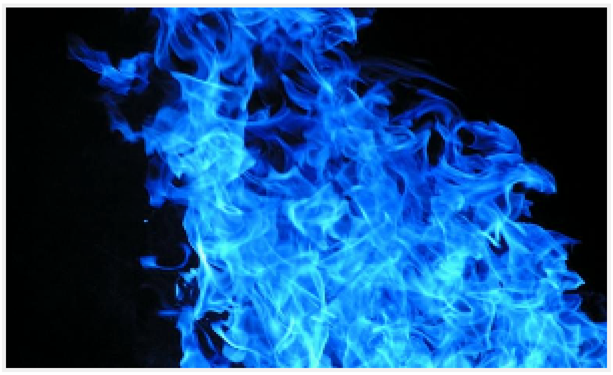

Fig2: HSV converted image of input image

3. **Non Processed Mask**: Shows the initial binary mask created by applying color-based thresholding on the HSV image. White pixels represent regions that match the defined HSV color range for fire.

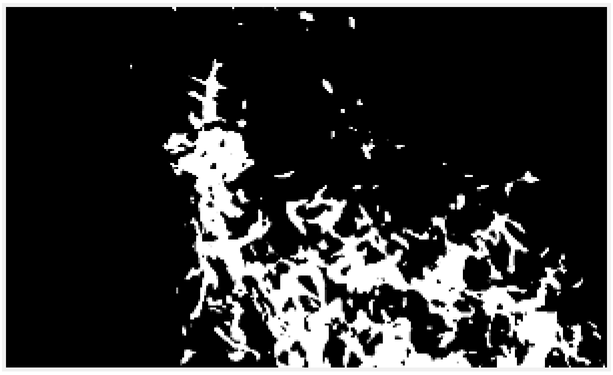

Fig3: Output after applying color-based thresholding

4. **Fire Mask after Morphological Closing**: Displays the binary mask after applying morphological closing. This operation helps to fill small holes within the detected fire regions, making the mask more uniform.

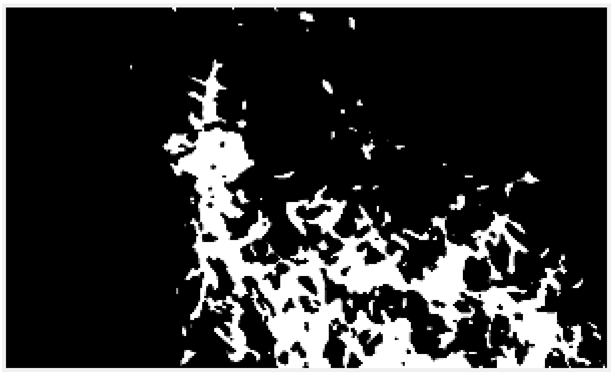

Fig4: Output after applying morphological closing

5. **Fire Mask after Morphological Opening**: Shows the binary mask after applying morphological opening. This operation removes small noise and artifacts from the mask, further refining the fire regions.

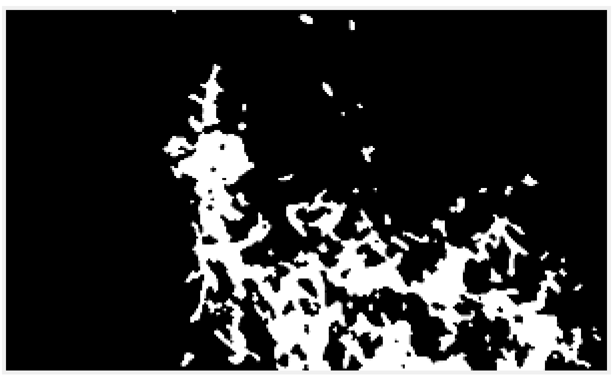

Fig5: Output after applying morphological opening

6. **Thresholded Fire Mask**: Represents the final cleaned binary mask after both morphological operations. This mask highlights the main fire regions with reduced noise and artifacts.

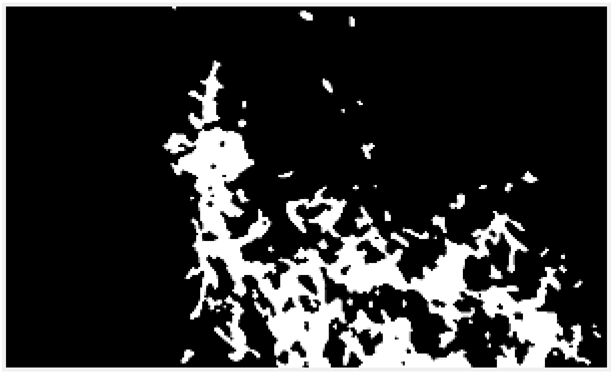

Fig6: Output after thresholding the fire mask

7. **Contour Detection**: Displays the original image with contours drawn around detected fire regions. These contours visualize the boundaries of potential fire areas, aiding in the identification process.

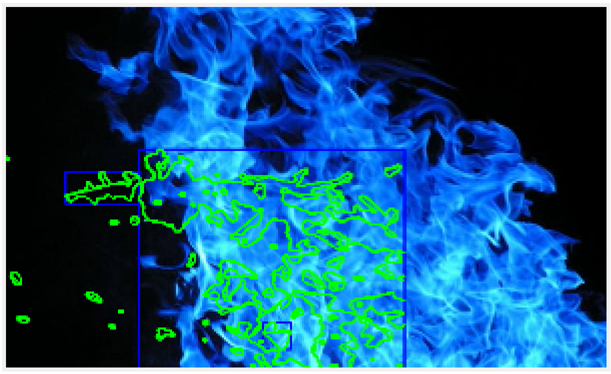

Fig7: Output after detecting the contours from the mask

8. **Output Image with Fire Detection**: Shows the final output image with bounding boxes around detected fire regions. The text "Fire Detected" in red indicates that fire was identified in the image. If no fire is detected, the text would appear in green as "No Fire Detected"

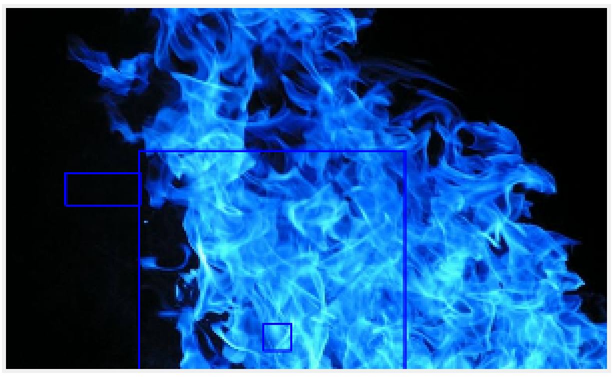

Fig8: Final fire detected output with boxed areas(fire area)

These output images provide a clear and sequential representation of the fire detection process, from color space conversion to the final result, enhancing transparency and understanding of the detection mechanism.

Conclusion

This project successfully demonstrates the application of basic image processing techniques for fire detection in images. By converting images to the HSV color space and applying color-based thresholding, the system effectively isolates potential fire regions. The subsequent morphological operations, including closing and opening, refine the binary masks to reduce noise and enhance the accuracy of fire region detection.

The user-friendly GUI, developed with Tkinter, allows users to easily load and process images, displaying each step of the fire detection process. This transparency not only makes the application accessible and understandable but also provides an educational tool for those interested in learning about image processing techniques.

The final results, with contours and bounding boxes around detected fire regions, offer a clear and visual representation of the detected fire areas. The color-coded output text further enhances the user experience by providing immediate feedback on the detection results.

While this project focuses on fundamental image processing methods, it serves as a robust foundation for more advanced fire detection systems. Future improvements could involve integrating machine learning models for more sophisticated detection or extending the application to handle real-time video streams.

Overall, this project achieves its objective of creating an efficient and transparent fire detection system using basic image processing techniques, highlighting the potential of these methods in practical applications.