Fire Detection System Using Yolov8 and Computer Vision

1. Introduction

The Fire Detection System is a computer vision-based solution that identifies the presence of fire in real-time video streams or uploaded video files. The system employs state-of-the-art machine learning techniques to detect fire using a trained deep learning model (fire.pt). In addition to fire detection, it integrates multiple advanced techniques such as optical flow analysis, edge detection, and texture contour analysis to enhance the reliability and accuracy of the detection process while minimizing false positives.

The system is designed to work both with video files and webcam inputs, providing a user-friendly interface with interactive options for file selection and webcam initiation.

2. Project Objectives

- **Real-time Fire Detection**: The primary objective is to detect fire in a video stream using a machine learning model and display alerts when fire is detected.
- **Advanced Image Processing**: Optical flow, edge detection, and texture contour analysis are incorporated to improve the accuracy of detection and reduce false alarms.
- **User Interaction**: A graphical user interface (GUI) is developed using Tkinter to allow users to either upload video files or use a webcam for detection.
- **Alert System**: When fire is detected, the system will show an alert message on the video feed.

3. System Overview

Key Components:

- **YOLO (You Only Look Once) Model**: A pre-trained fire detection model (fire.pt) is used to detect fire in video frames.
- **Optical Flow**: This technique calculates the movement of objects between two consecutive frames, helping in motion-based analysis.
- **Edge Detection**: Canny edge detection is applied to identify the boundaries of objects, aiding in clearer feature identification.
- **Texture and Contour Analysis**: This technique helps in analyzing the texture of images and drawing contours to further refine detection and reduce false positives.

- **Tkinter GUI**: Provides an easy-to-use interface to select a video file or use a webcam for live detection.

4. Methodology

The fire detection system operates in the following steps:

4.1 Model Selection

A pre-trained YOLO model (fire.pt) is used for fire detection. YOLO is an efficient deep learning architecture for real-time object detection that classifies and detects the presence of fire within a frame.

4.2 Pre-processing

- **Resizing**: Each frame is resized to a uniform size of 640x480 pixels for consistent processing and to reduce computational load.
- **Grayscale Conversion**: Each frame is converted to grayscale to simplify the image processing tasks (e.g., optical flow, edge detection).
- **Gaussian Blur**: A Gaussian blur is applied to smooth the image before edge detection, which helps in identifying key features more effectively.

4.3 Advanced Analysis

- **Optical Flow Analysis**: Optical flow is used to detect the movement between consecutive frames. This can help identify areas with unusual motion patterns that may indicate fire.
- **Edge Detection**: The Canny edge detector is used to highlight significant edges in the image, which are essential in distinguishing fire-related features from the background.
- **Texture and Contour Analysis**: Adaptive thresholding and contour analysis are applied to detect areas with unusual texture and shape characteristics that may be linked to fire.

4.4 Fire Detection

The YOLO model is applied to each frame of the video to detect fire. If fire is detected, a bounding box is drawn around the detected fire, and a confidence percentage is displayed. If the confidence level is greater than a threshold (50%), the system flags it as a potential fire.

4.5 Alert Mechanism

When fire is detected, an alert message ("Alert: Fire Detected!") is displayed on the video feed, warning the user about the potential danger.

4.6 User Interface

A graphical user interface (GUI) is created using Tkinter that allows the user to:

- Select a video file for fire detection.
- Open the webcam for live fire detection.

5. Implementation

5.1 Libraries and Dependencies

The system is built using the following libraries:

- **OpenCV**: For image processing tasks such as grayscale conversion, edge detection, and video handling.
- **Ultralytics YOLO**: For object detection, specifically fire detection using a pre-trained model.
- **cvzone**: For displaying text and bounding boxes on the video feed.
- **Tkinter**: For the graphical user interface, allowing users to interact with the system.

5.2 Code Structure

The code consists of the following key functions:

- select_video_file(): Opens a file dialog to allow the user to select a video file.
- open_webcam(): Initiates webcam-based fire detection.
- detect_fire(source): Handles the fire detection logic, processes video frames, and runs the detection algorithms.
- analyze_texture_contours(gray): Analyzes the texture and contours of the image to help improve fire detection accuracy.
- setup_gui(): Sets up the Tkinter GUI, providing buttons to select video files or open the webcam.

6. Results

The system performs real-time fire detection on video files or live webcam streams. When fire is detected, the system performs the following:

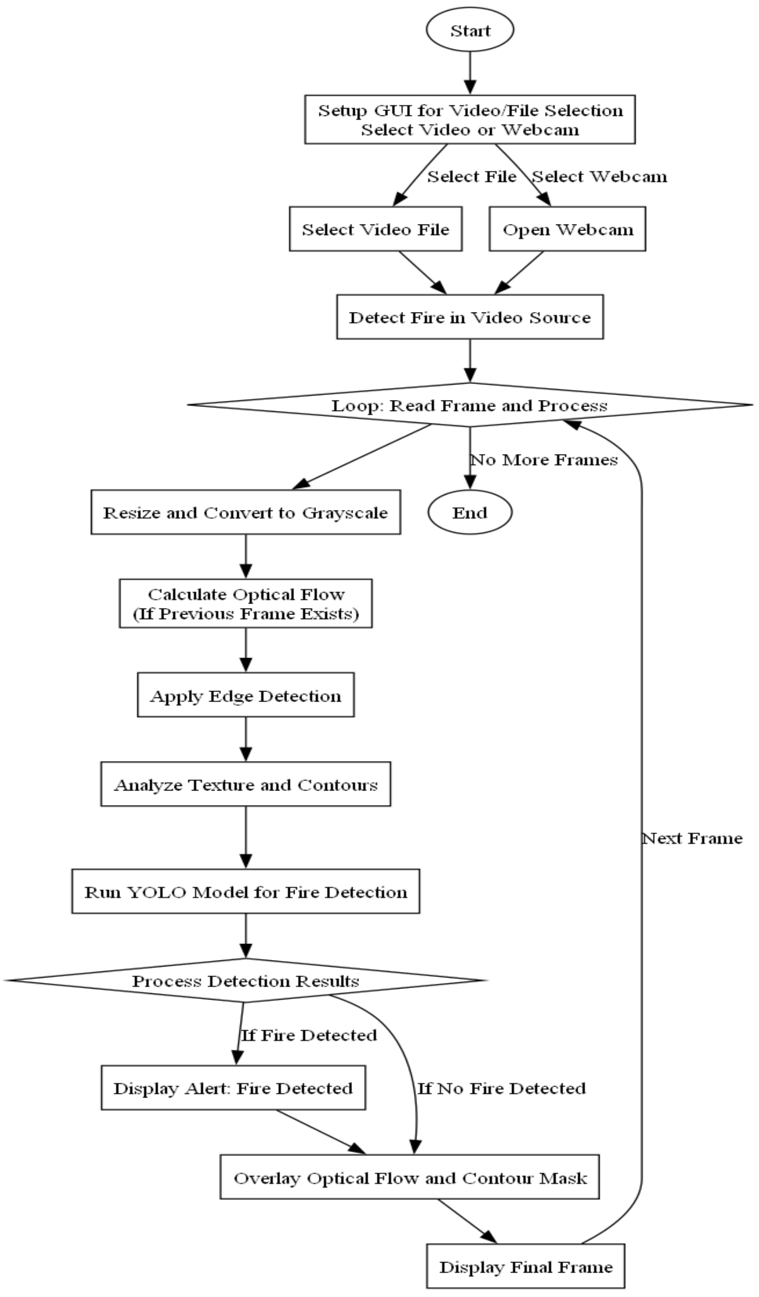
- Draws bounding boxes around detected fire areas.
- Displays the confidence level of the detection.
- Overlays optical flow and texture contour analysis on the frame.

- Triggers an alert message on the video feed if fire is detected.

Example Output:

- **Bounding Box**: A red bounding box is drawn around the detected fire.
- **Confidence Percentage**: The system displays the detection confidence level (e.g., "fire 85%").
- **Alert Message**: "Alert: Fire Detected!" is displayed on the top-left corner of the screen.

7. Flowchart


Start
Setup GUI for Video/File Selection
Select Video or Webcam
Select File
Select Webcam
Select Video File
Open Webcam
Detect Fire in Video Source
Loop: Read Frame and Process
No More Frames
End
Resize and Convert to Grayscale
Calculate Optical Flow
(If Previous Frame Exists)
Apply Edge Detection
Analyze Texture and Contours
Run YOLO Model for Fire Detection
Process Detection Results
If Fire Detected
If No Fire Detected
Display Alert: Fire Detected
Overlay Optical Flow and Contour Mask
Display Final Frame
Next Frame

8. Challenges and Improvements

Challenges:

- **False Positives**: Despite advanced texture and contour analysis, there may still be some false positives where non-fire objects are mistakenly detected as fire.
- **Lighting Conditions**: Variations in lighting conditions can affect detection accuracy, as the system may struggle to detect fire in low-light scenarios.

Future Improvements:

- **Model Optimization**: Using a more specialized fire detection model or retraining the existing model on a larger and more diverse dataset could improve accuracy.
- **Multiple Object Detection**: The system could be extended to detect multiple types of emergencies, such as smoke, gas leaks, or intrusions.

9. Conclusion

The Fire Detection System is a robust and efficient solution for real-time fire detection. By combining advanced image processing techniques such as optical flow, edge detection, and texture analysis with deep learning-based fire detection, the system can detect fire with high accuracy and provide timely alerts. The graphical user interface enhances the user experience by offering easy interaction with both video files and webcams.

10. References

- **YOLO (You Only Look Once)**: https://pjreddie.com/darknet/yolo/
- **OpenCV Documentation**: https://docs.opencv.org/
- **cvzone Documentation**: https://github.com/cvzone/cvzone