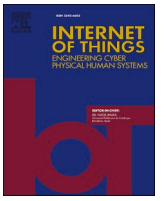
Internet of Things 28 (2024) 101402

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



Automated image-based fire detection and alarm system using edge computing and cloud-based platform

Xueliang Yang, Yenchun Li, Qian Chen *

School of Engineering, University of British Columbia (Okanagan Campus), 3333 University Way, Kelowna, BC V1 V 1V7, Canada

ARTICLE INFO

Keywords:
Automated fire detection
Thermal Imaging
Cloud and edge computing platform
YOLO

ABSTRACT

To tackle the increasing wildfire challenges, this paper presents an automated image-based fire detection and alarm system utilizing edge computing and a cloud-based platform, specifically designed for urban building fire detection. The system captures both RGB and infrared images from thermal cameras and employs existing computer vision techniques to detect fire characteristics such as flames and smoke. By integrating edge computing, the system minimizes latency to enhance the accuracy of fire detection and alarm activation. The cloud platform supports extensive data storage, analysis, and remote monitoring, which can ensure data accessibility and scalable data management. The proposed system descriptions include a detailed system architecture design, data collection, and the selection and application of detection algorithms that leverage both RGB and thermal image data for fire detection. Using the campus building and surrounding risk-prone areas as a testbed, the proposed system demonstrated desired fire detection capabilities and a robust solution to quickly identify and respond to fire incidents within the urban area. The proposed system functionalities can be scaled and adapted to other fire risk-prone areas.

1. Introduction

In recent years, wildfires have increased the frequency, intensity, and duration globally. Since 1990, on average, Canada has experienced approximately 7500 wildfires each year [1]. Hotter and drier summers, combined with reduced snow cover, make vegetation more flammable. With population growth and urban expansion, more people are moving closer to forested areas and wildlife habitats, heightening the threat to human lives and property. Wildfires also have imposed a significant economic impact across Canada and particularly in BC, affecting tourism, agriculture, forestry, and infrastructure, with losses potentially reaching billions of dollars. Additionally, wildfires not only destroy forests and wildlife habitats but also deteriorate air quality, posing health risks to humans. Post-fire landslides and soil erosion can have long-term effects on ecosystems [2]. In 2018, BC reported approximately 2100 wildfires, affecting over 1350,000 hectares. In 2021, the number of wildfires decreased slightly to about 1600, but the affected area still reached 870,000 hectares. In 2023, BC experienced its worst wildfire season on record, reporting 2245 wildfires that burned over 3000,000 hectares of land, with nearly 200,000 people forced to evacuate [3]. These data indicate that the frequency and severity of wildfires in BC are increasing, significantly impacting communities, ecosystems, and the economy, highlighting the importance of strengthening wildfire resilience and prevention measures.

Fire detection and monitoring, either in an urban or forest setting, is a complex and crucial task, for which various advanced remote

* Corresponding author.
E-mail address: qian.chen@ubc.ca (Q. Chen).

https://doi.org/10.1016/j.iot.2024.101402

Available online 10 October 2024
2542-6605/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

sensing and Internet of Things (IoT) technologies have been developed and applied [4]. Satellite-based remote sensing is at the core of these efforts, with satellite systems such as NOAA-AVHRR providing large-scale continuous monitoring and detailed imagery of fire conditions, smoke, and cloud cover. Ground-based camera systems utilize visible light and infrared sensors for real-time monitoring of local areas, capable of detecting new fires during both day and night. Wireless sensor networks deploy numerous nodes throughout forests to monitor environmental parameters such as temperature and humidity. IoT solutions, like the "electronic nose" sensors developed by Dryad Networks, can detect gases associated with wildfires in their early stages [5]. Housing construction stakeholders have started to explore automation and digital technologies to build fire-resilient homes [6]. These technologies complement each other, forming a multi-layered, comprehensive forest fire monitoring system that greatly enhances early fire detection and rapid response capabilities, providing robust technical support for effective forest fire prevention and control.

To improve local area fire detection, deep learning and computer vision methods are explored by many researchers. One method employs Convolutional Neural Networks (CNN) to analyze video frames for detecting flames and smoke, effectively identifying fire events based on visual patterns. A study demonstrated that a multi-scale prediction model using CNNs significantly enhances fire detection accuracy by capturing features at various scales [7]. Another approach integrates Long Short-Term Memory networks with variational autoencoders to improve detection sensitivity and reliability by learning temporal patterns and distinguishing fire-related anomalies [8]. Furthermore, a technique based on multi-scale feature extraction and implicit deep supervision has been shown to be efficient in detecting fire by leveraging channel attention mechanisms to focus on relevant features [9]. Additionally, the use of aerial imagery combined with deep learning models provides precise fire detection capabilities. This approach analyzes visual data from drones to identify and monitor fire events in real-time, enhancing early warning systems and response strategies [10].

A heavy focus in the current body of literature is on algorithms designed for fire pattern recognition, however, a comprehensive fire detection and alarm system has been inadequately studied due to the challenges in data and technology integration. To address this knowledge gap, this study proposes a scalable and automated fire detection and alarm system that utilizes edge computing and a cloud-based platform, specifically designed for urban local area fire detection. The system captures both RGB visible and infrared images from FLIR thermal cameras (e.g., model A700), employing existing computer vision techniques to detect fire characteristics such as flames and smoke using images collected from the cameras. By integrating edge computing, the system minimizes latency, enhancing the availablity of fire detection and alarm activation. The cloud platform supports extensive data storage, analysis, and remote monitoring, ensuring scalability and real-time data accessibility. It is important to note that the purpose of this paper is not to innovate on fire detection algorithms, but rather to focus on the system's architecture and integration. A key spotlight of this research is the tight integration of this research and the Campus Facility Operation staff feedback to deploy and test the proposed system in a real-world context for the university campus fire detection to improve campus climate resilience. By using the campus as a testbed (a living lab), this study aims to prototype the automated information communication system to enhance fire detection capabilities to provide a robust solution to quickly identify and respond to fire incidents within the campus, thereby improving overall safety and minimizing potential damage to campus facilities. The proposed system can be easily adapted and scaled to different contexts of fire detection in the urban setting due to the modular system architecture consisting of both of cloud computing and edge computing processes.

The rest of the paper is structured as follows: Section 2 reviews the current methods for smart fire detection and monitoring. Section 3 provides a detailed description of the methodology proposed in this study, including system architecture design, data collection, processing procedures, and the selection and application of detection algorithms. Section 4 demonstrates the feasibility and effectiveness of the proposed system through its deployment on campus (UBC Okanagan Campus). Section 5 discusses the advantages of the proposed system, such as real-time performance, accuracy, and scalability and points out its limitations for future improvement. Finally, Section 6 summarizes the research findings and provides an outlook on future work.

2. Literature review

2.1. Sensing technologies for fire detection

Over the past decade, fire detection systems have undergone technological iterations and updates, and scholars have conducted research on multi-sensor fusion methods based on Wireless Sensor Networks (WSNs) and the Internet of Things (IoT), which have shown great potential in reducing false alarms and improving fire sensitivity. Four primary types of fire detection sensors are utilized: heat, smoke, gas, and optical detectors [11]. Each type has its own unique strengths and limitations, making them suitable for different applications and environments.

Heat Fire Detectors. Heat fire detectors primarily detect fires based on temperature changes. Common heat detectors include fixed temperature sensors and rate-of-rise temperature detectors. The former detects fires when the temperature exceeds a certain threshold, while the latter identifies fires based on an abnormally rapid temperature increase. Gottuk et al. used conventional smoke detectors combined with carbon monoxide sensors to develop multi-signature alarm algorithms for improved fire detection [12]. Zervas et al. explored a multi-sensor approach, including heat detectors, to create a new method for early fire detection [13]. The temperature sensors were designed to be simple, low-cost, and easy to maintain. However, their drawbacks include low sensitivity and inability to adapt to environments with large temperature fluctuations.

Smoke Fire Detectors. Smoke fire detectors detect fires by sensing smoke concentration. These are the most common fire detectors in daily life, as they are relatively sensitive to smoke. However, their sensitivity also makes them susceptible to environmental influences. For instance, high dust concentration or humidity can cause false alarms. Bukowski et al. conducted a comprehensive performance analysis of different residential smoke alarm technologies in various fire settings [14].

Gas Fire Detectors. Gas fire detectors detect fires by measuring the concentration of toxic gasses, such as carbon monoxide,

produced during combustion. As a result, gas fire detectors are highly susceptible to performance degradation due to air movement. Xiao et al. reviewed the applications of carbon nanotube-based CO sensor for early fire detection [15]. Su et al. reviewed fire detection methods based on multi-sensor data fusion, including the use of gas detectors [16].

Optical Fire Detectors. Optical fire detectors utilize computer vision and image processing technologies to identify fire characteristics such as flames, smoke, and thermal radiation. This approach enables long-distance and large-area fire monitoring, making it particularly suitable for open spaces and large buildings. Toreyin et al. presented a computer vision-based method using optical sensors for real-time fire and flame detection through image processing techniques [17]. Chen et al. explored the use of image processing techniques with optical sensors for early fire detection through smoke detection algorithms [18]. However, compared to traditional detectors, camera-based systems offer the advantages of wide coverage and high visualization, providing intuitive information about the fire's location and development. However, this method also faces challenges such as false alarms in complex environments, sensitivity to lighting conditions, and potential privacy issues in some cases. Despite these challenges, with the advancement of artificial intelligence and deep learning technologies, optical fire detectors are continually improving in accuracy and robustness, currently seen as an increasingly important component of modern fire warning systems.

Different detectors are suitable for different application scenarios. For instance, optical fire detectors are more suitable for urban environments, such as buildings, offices, and public places, as well as indoor areas, or outdoor areas like campuses or streets [19]. This is because urban areas often have an extensive network of cameras, making it convenient to collect image and video data for fire detection without the need for additional sensors. Indoor and urban outdoor spaces have more controllable lighting and environmental conditions, allowing computer vision algorithms to detect fire and smoke more accurately [20]. In contrast, remote forest fire detection faces challenges due to the lack of existing camera infrastructure, and large forest areas are difficult to cover solely with cameras. Moreover, diverse lighting, weather, and seasonal changes can interfere with the accuracy of vision-based systems. Therefore, multi-sensor data fusion, intelligent analysis techniques, and technologies such as unmanned aerial vehicles (drones) and the Internet of Things (IoT) can be employed to enhance the speed, accuracy, and reliability of fire detection in various scenarios. [21].

2.2. Computer vision techniques used for fire detection

Because of the significant color characteristics and temperature features present during a fire, current machine vision-based fire detection technologies are primarily divided into two categories: visible image fire detection and infrared image fire detection.

There has been a lot of research on visible image fire detection. For example, Sharma et al. developed a deep convolutional neural network approach for fire detection in images, demonstrating improved accuracy and reduced false alarms compared to traditional methods [22]. Liu et al. conducted a comprehensive survey of computer vision-based fire detection methods, covering traditional image processing techniques and advanced deep learning approaches [23]. Jain et al. reviewed machine learning applications in wildfire science and management, including the use of computer vision techniques for fire detection in satellite and aerial imagery [24]. In the system development work, Wu et al. created a dataset for fire and smoke object detection to provide a valuable resource for developing and evaluating camera-based fire detection algorithms [25]. In a similar vein, Jin et al. presented an overview of video fire detection methods based on deep learning, discussing the datasets, methodologies, and future directions in the field [26]. Along a similar line, Khan et al. introduced a novel dataset and deep transfer learning benchmark for forest fire detection using visible light images, enhancing the performance of camera-based fire detection systems [27]. Jiang et al. proposed an attention mechanism-improved YOLOv7 object detection algorithm, which can be applied to fire detection tasks using camera-captured images [28]. Gragnaniello et al. proposed a novel application scenario-based taxonomy, including short-range low-activity scenarios, long-range low-activity scenarios, short-range high-activity scenarios, and long-range high-activity scenarios, to assess the performance of various fire detection methods on different datasets [29].

Early infrared imaging technology was expensive, so fire detection methods primarily focused on visible light images. As the cost of infrared cameras gradually decreased, they became more affordable and research was conducted on the application of infrared technology in fire detection. Infrared images highlight the thermal distribution of objects within the field of view, effectively eliminating light source interference and distinguishing between objects with different temperatures and flames. Infrared fire detection technology based on traditional methods started early and has developed relatively maturely. For example, Qin et al. [30] applied a 940 nm narrow-band filter to the camera to obtain infrared images by analyzing the solar spectrum. Then, a flame threshold selection method was used to obtain binary images, and the similarity of consecutive video frames and the roundness of contours were used for the final judgment. To automate the detection workflows, Yuan et al. [31] proposed a detection method based on infrared images from drones. They first used a histogram segmentation method to identify suspected fire areas in the infrared images and then analyzed these areas using the optical flow method to determine if a fire had occurred. Similarly, Kim et al. [32] proposed a video fire alarm system based on infrared images. The system incorporated a flame recognition algorithm that first identified candidate flame areas in the infrared images and then judged whether it was a flame based on brightness changes within the area. Shekhar et al. [33] used four different wavelength infrared sensors to detect flames and employed algorithms such as auto-correlation to improve the accuracy of flame detection, aiming to prevent false alarms caused by high-temperature refractory materials and avoid the inability to detect flame flicker due to saturation effects.

In recent years, with the rapid development of deep learning technology, many scholars have applied deep learning to fire detection based on infrared technology. For example, Aue et al. [34] proposed an infrared fire alarm system based on neural networks, combining infrared image information with meteorological and geographical information to detect and alert forest fires. Wang et al. [35] used dual-band infrared vision technology to eliminate light source interference, extract flame features, and establish a BP neural network model to identify flames. Li et al. [36] proposed a dynamic texture recognition method based on deep neural networks. They

divided videos into spatiotemporal blocks, used local binary patterns (LBP) to extract features, and employed a compressed sensing-based infrared thermal imaging feature extraction algorithm to achieve early fire detection. Focusing on flame features, Aslam et al. [37] used dynamic infrared fire image textures to calculate flame roundness and then performed image recognition based on infrared flame features through a convolutional neural network. Wang et al. [38] proposed a nine-layer convolutional neural network (IRCN) to replace traditional empirical manual methods for extracting infrared image features. The extracted features were then used to train a linear support vector machine, achieving fire detection and enabling fast detection on ordinary infrared surveillance cameras. Li et al. proposed an automatic fire detection method based on drone infrared images. They first used an improved YOLOv3 algorithm for target detection in infrared images, then combined morphological processing and dynamic threshold segmentation techniques to extract fire features, and finally used a support vector machine (SVM) classifier for fire detection [39]. To generalize the detection capacity, Zhang et al. studied an infrared image fire detection algorithm based on a convolutional neural network. They used an improved Faster R-CNN network for feature extraction and target detection in infrared images and enhanced the model's generalizability through transfer learning and data augmentation techniques, achieving high-precision fire detection [40]. Akhloufi et al. proposed a deep learning-based wildfire detection method for drone images. They used two deep learning models, YOLOv3 and RetinaNet, for fire detection in infrared and visible light images, and improved detection performance, especially for small fire targets, through multi-scale feature fusion and attention mechanisms [41]. To effectively improve detection accuracy and robustness, Liu et al. combined deep learning and handcrafted features to propose a wildfire detection method for infrared images. They first used an improved VGG16 network to extract deep features, then combined these with handcrafted color and texture features, and finally used a random forest classifier for fire detection [42]. Wu et al. proposed an adaptive threshold-based deep learning method for fire and smoke detection in multispectral images. They used an improved YOLOv3 network for feature extraction and target detection in infrared and visible light images and optimized detection results through an adaptive threshold strategy, enhancing the system's detection performance especially in complex environments.

Infrared images lack color and texture information and rely solely on temperature information to determine the presence of flames. Although single-channel infrared flame detection methods can avoid missed detections to some extent, they may produce false alarms under thermal interference.

2.3. Knowledge gap

The integration of advanced sensing technologies and computer vision techniques has significantly improved the capabilities of fire detection systems. Section 2.1 has reviewed the various types of fire detection sensors, such as heat, smoke, gas, and optical detectors, each with its unique advantages and tailored for specific environments. Section 2.2 has detailed the evolution of computer vision in fire detection, from traditional image processing to the sophisticated application of deep learning algorithms, enhancing the accuracy and reducing false alarms in detecting fires through visual and infrared images. Despite these technological advancements, there remains a significant gap in the development of a workable fire detection systems in an urban setting that are scalable and suitable for real-world engineering applications. Current research has predominantly focused on refining detection technologies and algorithms in isolation, without adequate consideration for the practical challenges of urban fire detection scenarios in areas that are prone to wildfire threats. In order to monitor and detect fire in dense population areas close to forests such as easy adoption of certain fire detection system on campus, the existing methods often struggle to meet the demands of large-scale data acquisition and analytical processing, which are critical for effective fire detection and response in urban environments. In this scenario, it is important to consider not only the detection algorithm but also how to implement simultaneous monitoring of target building complexes through multiple cameras and different image types. As discussed in the study [43], combining diverse sources of information and technologies is essential to enhancing detection accuracy and performance. Beyond fire detection in images, there is a need to provide an integrated platform for users to further review and confirm alerts. Currently, there has been limited mature research in this area. This paper proposes a framework that integrates edge and cloud computing to build a fire detection platform to safeguard the urban environment and buildings.

3. Methodology

In this paper, an automated image-based fire detection and alarm system is proposed, which includes cameras, a cloud-based dashboard and detection services, as well as edge devices. Thermal cameras are used to capture both RGB and infrared images. A cloud-based fire detection service and dashboard are developed to provide surveillance and automatic fire detection capabilities. The cloud services are designed to scale according to the workload and prevent single points of failure. To mitigate the impact of network fluctuations on image collection and analysis, edge devices are employed to directly connect the cameras deployed in important areas and capture images for fire detection. This approach enhances the reliability of monitoring critical areas, thereby improving the overall robustness of the system [44].

3.1. Overall system architecture

In this research, a cloud-edge collaborative architecture is used to construct an urban building fire detection system, which is shown in Fig. 1. The cloud side adopts a distributed micro-service architecture, supporting large-scale camera data collection, synchronized analysis of various detection algorithms, and a user-friendly dashboard. The edge side works with cameras and sensors for local data collection and analysis, performing low-latency detection and analysis. The details of the architecture are introduced in the following

parts of this section. The cloud-edge collaborative architecture is adopted in this system to address the challenges of scalability and availability in fire detection for urban buildings with a large number of cameras. According to Wang et al. [45], service systems benefit from modular and distributed designs as they enhance flexibility and fault tolerance, especially in complex environments like enterprise information systems. Applying this concept, the cloud side provides scalable data processing and algorithm synchronization, while the edge side ensures low-latency detection by processing data locally. This architecture enables efficient handling of large-scale camera networks and ensures the system's robustness and responsiveness.

3.2. Cloud-based architecture

The fire detection system studied in this project is broadly divided into four layers: the device layer, data collection layer, fire detection layer, and user interface layer, as is shown in Fig. 2.

The device layer currently supports the integration of the FLIR thermal camera model A700, which is capable of multiple data transmission protocols including RTSP, HTTP, MQTT, and MODBUS. This layer is critical as it forms the interface between the physical environment and the digital system, capturing raw data that is essential for fire detection. The use of the FLIR Thermal Camera enhances the system's ability to detect temperature anomalies that may indicate the presence of fire, even in low-visibility conditions.

The data collection layer includes functionalities such as image and video capture, service registration, and load balancing. Image and video capture employs RTSP streaming to capture dynamic footage, periodically taking snapshots and publishing them to the Kafka message queue, while periodically uploading video clips to AWS S3 storage. Service registration involves registering the data collection service with Zookeeper registry center, monitoring the online status of other members in the data collection service cluster. The load balancing function uses consistent hashing, leveraging the list of online members and the camera ID to calculate which data collection service a camera should be assigned to. Even in the event of changes in the data collection service cluster, such as program crashes or system instance failures, it can dynamically reassign the affected cameras to other data collection services to continue capturing data.

The fire detection layer includes fire detection services, load balancing, and the integration of various algorithms. The current fire detection service uses YOLO v8, based on computer vision, to detect flames and smoke. Detected alerts are published to Discord, allowing subscribed users to view alert information and on-site images. Alert information is also stored in MongoDB, enabling users to view historical alerts from the web interface. The load balancing method differs from that of the data collection layer. It is based on Kafka partitioning. Services using the same detection algorithm subscribe to Kafka image topics with the same Group Id. This way, when subscribing to and retrieving on-site images published by the data collection layer from the Kafka message queue, detection algorithms within the same group can use Kafka's sticky partitioning algorithm to distribute messages (on-site images) from different Kafka partitions.

The user interface provides functionalities to query historical alert records from MongoDB, view historical videos from AWS S3 and manage users. Discord is also integrated into this design, anyone who subscribes to the fire alert channel can receive fire alarm in real time.

3.2.1. Service design

(1) Data Collection Service

During the design phase of the Data Collection Service (DCS), shown in Fig. 3, high availability and scalability were the primary considerations. In the proposed design, RTSP Stream is used to capture images and videos from cameras. The number of cameras that can be concurrently captured by each server on the cloud platform is limited, so multiple servers are required to run the DCS. The number of servers can be calculated by dividing the total number of cameras by the number of cameras each server can handle

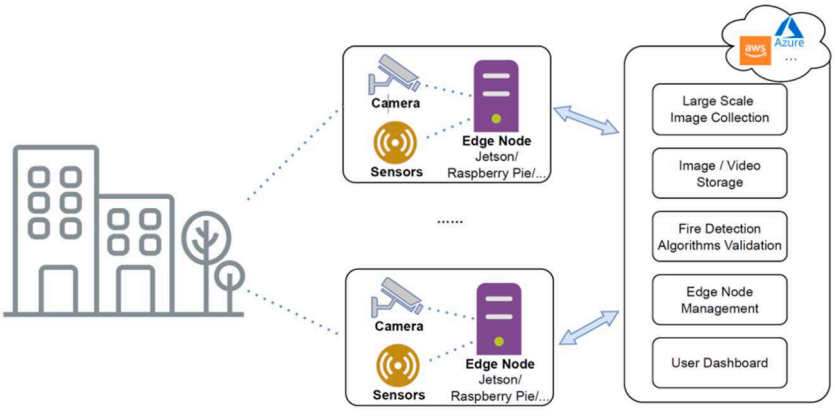


Fig. 1. Architecture of edge-cloud fire detection.

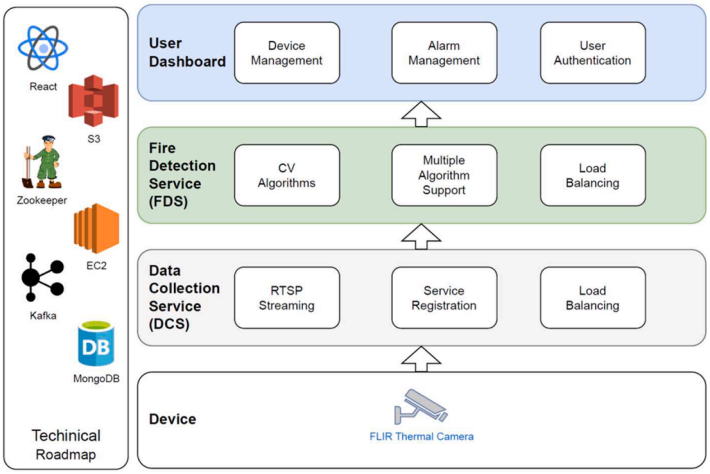


Fig. 2. Architecture of cloud-based fire detection.

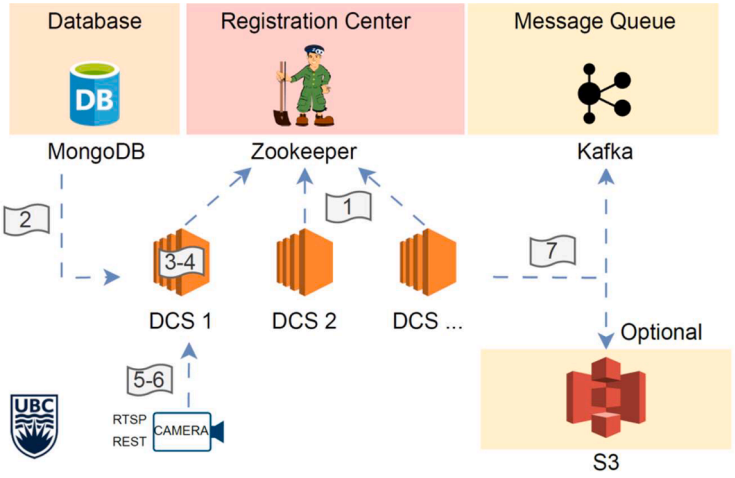


Fig. 3. Data collection service design.

concurrently. On each server, the calculation of the maximum number of threads should also consider the implementation method of data collection. For RTSP streaming, the most common approach is to use one thread per camera. To ensure reliable collection, the maximum number of threads should not exceed twice the number of cores minus one. This prevents frequent context switching and reduces the additional overhead caused by excessive concurrency, which would otherwise degrade system performance. Based on the above estimation, to achieve fire detection on the UBC Okanagan campus with a potentially large number of cameras, a scalable architecture is necessary. As the number of cameras increases, additional DCS instances can be added to expand the collection capacity. This imposes higher requirements on the design of the DCS distributed architecture. This paper introduces the design from three aspects: processing flow, service registration mechanism, and load balancing.

The flowchart shown in Fig. 4 outlines the steps involved in the DCS for fire image collection. The detailed description of each step is provided as follows.

Step1 - Register Data Collection Service. The first step involves registering the DCS in Zookeeper. The details of this registration is illustrated in Fig. 5.

Step2 - Watch the status of the group. Each DCS node will watch the group information registered in Zookeeper at a time interval of 5 s.

Step3 - Load balancing using Consistent Hashing. Distribute the workload evenly among the available servers using consistent hashing to ensure efficient load balancing.

Step4 - Initiate a thread pool. Initialize a pool of threads to handle the concurrent data collection tasks.

Step5- Initiate RTSP streaming. Start the RTSP streaming for each camera in a separate thread. Capture RGB images and videos from the camera at 5-second intervals. The time interval can be customized according to the config.

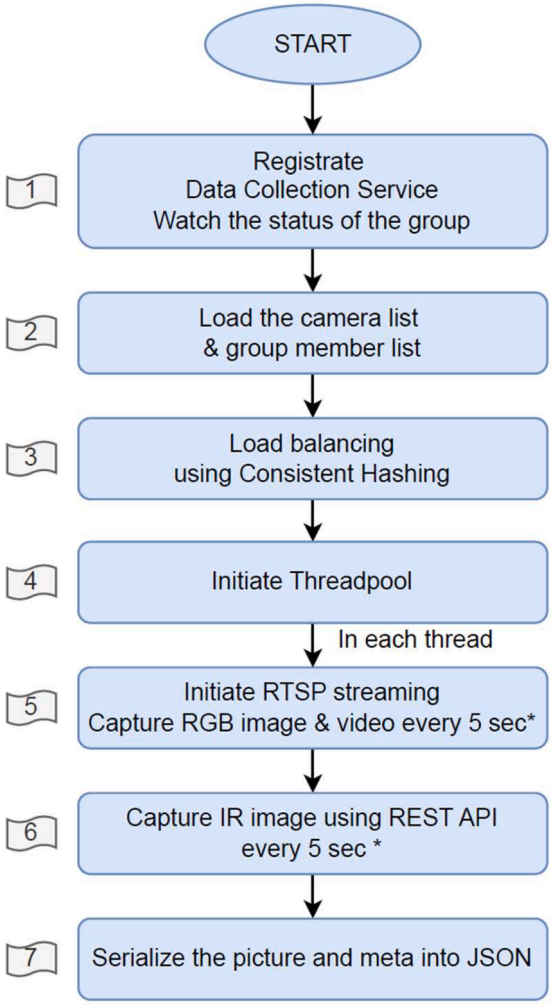


Fig. 4. Flow chart of DCS.

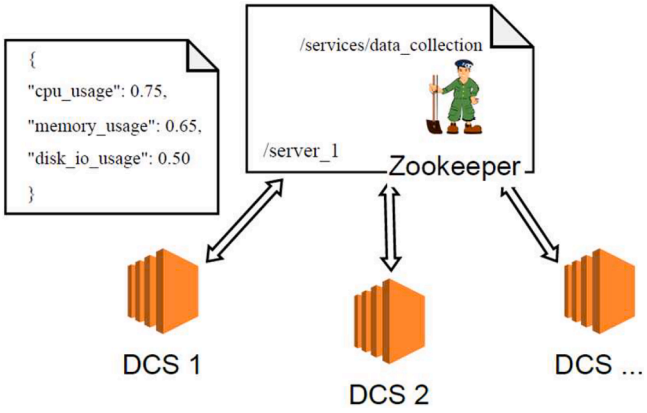


Fig. 5. DCS registration in Zookeeper.

Step6 - Capture IR image using REST API every 5 s. Using a REST API, capture infrared (IR) images from the camera at 5-second intervals. The time interval can be customized according to the configuration.
Step7 - Serialize the picture and meta into JSON. Convert the captured images and their metadata into JSON format and publish the image JSON into the topics of RGB and IR image respectively. This structured approach ensures efficient data collection, processing, and load balancing across multiple servers, enhancing the reliability and scalability of the fire detection system.

To ensure high availability and load balancing of the system, we use Zookeeper as the service registry center. With Zookeeper, we can dynamically register and manage instances of data collection services and monitor the resource usage of each service instance in real time. When each data collection service instance starts, it registers its information in Zookeeper, including CPU, memory, and disk I/O usage. This information is stored under a specific path, such as /services/data collection.

Using Zookeeper's CLI tool, we can view the registered service nodes. For example, if there are three server nodes, server_1, server_2, and server_3, executing the command /services/data collection in Zookeeper's zkCli tool will display [server_1, server_2, server_3]. Using the get command to retrieve the registration information of each node, we can see the performance parameters of each server node, such as CPU, memory, and disk I/O usage.

Here is an example of retrieving the information of the server 1 node: get/services/data collection/server_1. The retrieved registration information would be: "{"cpu usage": 0.75, "memory usage": 0.65, "disk io usage": 0.50}". The load balancing method for distributing image acquisition tasks in the data collection layer of this project design is shown in Fig. 5.

Consistent hashing is used to allocate image collection tasks to DCS [42]. We first initialize a hash ring and map DCS nodes onto it according to their hash values. Each physical server is assigned multiple virtual nodes to achieve load balancing. When an image collection task is to be assigned, the hash value of the camera ID is computed, and the task is allocated to the first virtual node on the hash ring with a hash value greater than or equal to this computed value. If no such virtual node is found, the task is assigned to the first virtual node on the ring. Dynamic adjustments to the number and distribution of virtual nodes are made based on performance metrics and load data, ensuring effective load balancing and efficient task distribution across the system.

(2) Fire Detection Service

The Fire Detection Service (FDS) also registers its service through Zookeeper and retrieves serialized image strings from the message queue Kafka. After deserializing these strings to obtain images, it uses fire detection algorithms to identify fire and smoke in the images and provide the coordinates of the bounding box. The overall design of the FDS is shown in Fig. 6.

The design of the FDS focuses on two main aspects: The system should be able to support multiple types of fire detection algorithms. Another one is that the system needs to handle increased detection loads as the number of cameras increases, ensuring scalability. By leveraging Kafka's features, different algorithms can be used simultaneously for fire detection. For example, if there are two FDS instances, there are two scenarios: both instances run the same algorithm or both instances run different algorithms. In the first scenario, where both instances run the same algorithm, if the configuration files of these two instances have the same group name, the instances will receive data from different partitions of the subscribed image topic. Together, they will cover all the images in that topic. If the configuration files have the same group names, each of the instances will receive all the data from the subscribed image topic. Using this characteristic, a distributed fire detection system can be constructed, where adding more nodes can linearly increase detection efficiency.

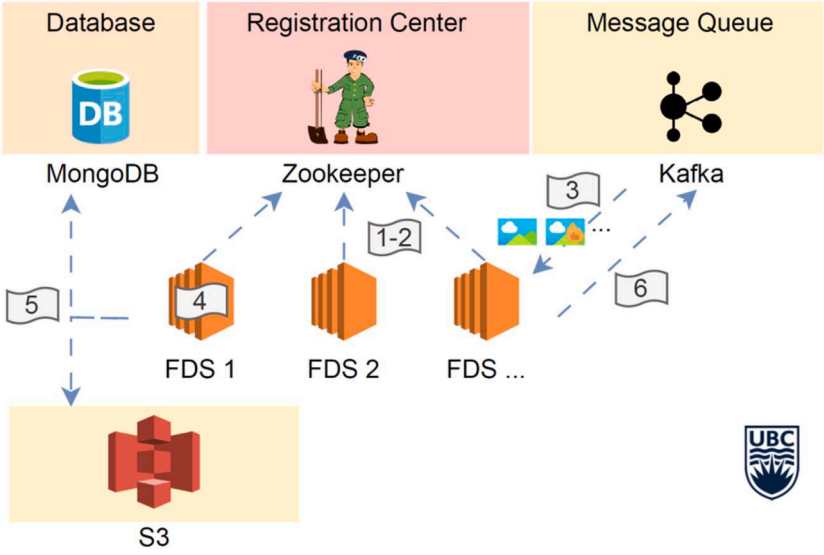


Fig. 6. Fire detection service design.

The flowchart shown in Fig. 7 illustrates the steps involved in the FDS process. A detailed description of each step is provided as follows.

Step1 - Register Fire Detection Service (FDS). The first step involves registering the fire detection service with the central system. Monitor the status of the group to ensure that all necessary services are operational and that the system is functioning correctly.
Step2 - Load balancing using Consistent Hashing [46]. Distribute the workload evenly among the available servers using consistent hashing. This ensures that the system handles the incoming data efficiently and distributes the processing load to avoid bottlenecks. The load balancing method is described in 2.4.2.
Step3 - Consume the images from Kafka. Retrieve the images sent by the data collection service from Kafka. Kafka acts as the messaging system that brokers the data between producers (DCS) and consumers (FDS).
Step4 - Detect fire in the images. Analyze the consumed images to detect the presence of fire. This step can use diverse algorithms, including image processing, machine learning, or deep learning techniques, to accurately identify fire-related anomalies in the images.
Step5 - Save alerts in the database and AWS S3. Once a fire is detected in an image, save the alert details in a database for quick access and analysis. Additionally, store the related images and metadata in AWS S3 for long-term storage and retrieval.
Step6 - Send the alerts to the Kafka or email. Notify the relevant stakeholders about the detected fire. This can be done by sending alerts back to Kafka for further processing or by directly sending email notifications to the concerned authorities.

This structured approach ensures that the FDS can efficiently monitor and detect fires by leveraging Kafka for data distribution, consistent hashing for load balancing, and various algorithms for image analysis. The system is designed to be scalable and reliable, ensuring timely alerts and appropriate storage of data for future reference.

The load balancing of FDS is primarily based on Kafka's consumer Sticky Assignor strategy. Kafka consumer groups need to assign partitions of topics to various consumers within the group. Common partition assignment strategies include Round-Robin Assignor, Range Assignor, and Sticky Assignor. The Sticky Assignor strategy is adopted in this project to aim for stable partition assignments for

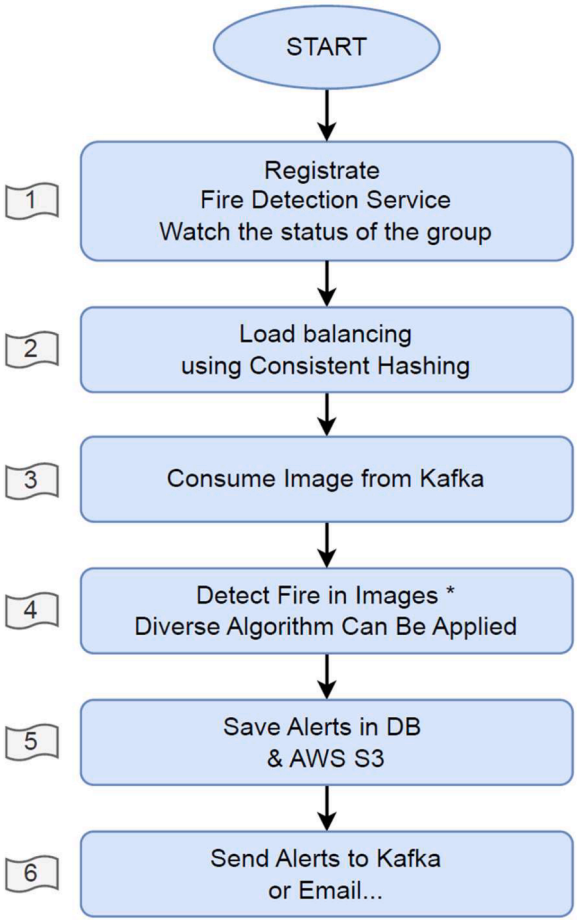


Fig. 7. Fire detection service workflow.

each consumer, thereby reducing the frequency of partition migrations during rebalancing and improving system stability and performance.

The Sticky Assignor strategy has two main objectives: firstly, to minimize partition reassignments by attempting to maintain existing bindings between consumers and partitions when group membership or subscriptions change; secondly, to balance the distribution of partitions as evenly as possible among consumers. The assignment process proceeds as follows.

Step 1 - Initial assignment. During the first assignment, partitions are initially assigned to consumers based on topic subscriptions and the number of consumers, using either a round-robin or range assignment strategy.
Step 2 - Sticky adjustments. When a new consumer joins or an existing consumer leaves, efforts are made to maintain the existing partition assignments unchanged. If reassignment is necessary, adjustments are made with minimal changes while preserving the maximum number of unchanged partition assignments.
Step 3 - Assignment process. First, compute the current assignment which is to determine the current partition assignment status. Then, Generate an initial assignment plan based on round-robin or range strategies and adjust the preliminary assignment plan while attempting to preserve existing partition bindings, achieving both stickiness and balance.
Step 4 - Balancing assignment. In the final assignment result, ensure that the number of partitions assigned to each consumer is as balanced as possible. If a consumer is assigned more partitions than the average, some partitions are reassigned to other consumers until balance is achieved.

By using the Sticky Assignor strategy, stability in partition assignments is maintained, reducing the overhead caused by partition reassignments such as consumer data reading and connection establishment. This minimizes the frequency of partition rebalancing and enhances the overall performance and stability of the system.

3.2.2. Data structure

As shown in Fig. 8, the data flow can be described in a sequential manner. The data between DCS and FLIR cameras are transferred in RTSP protocol, which is a widely used protocol for video surveillance and streaming media services, providing a flexible and efficient solution for real-time audio and video transmission. The collected images are serialized into JSON strings and published into Kafka whose format is shown in Appendix A. The video footage is stored in an Amazon S3 bucket called “fire-clips”. The Kafka topics are subscribed by FDS nodes, the format of the JSON string is the same as described in Appendix B. Once fire or smoke is detected in an image by FDS, there will be an alarm sent into both Kafka and MongoDB. The format of this alarm can be seen in Appendix A. The images which triggered the alarm are stored in an S3 bucket called “fire-clips”. The dashboard of this system can present the images and videos in AWS S3 and MongoDB.

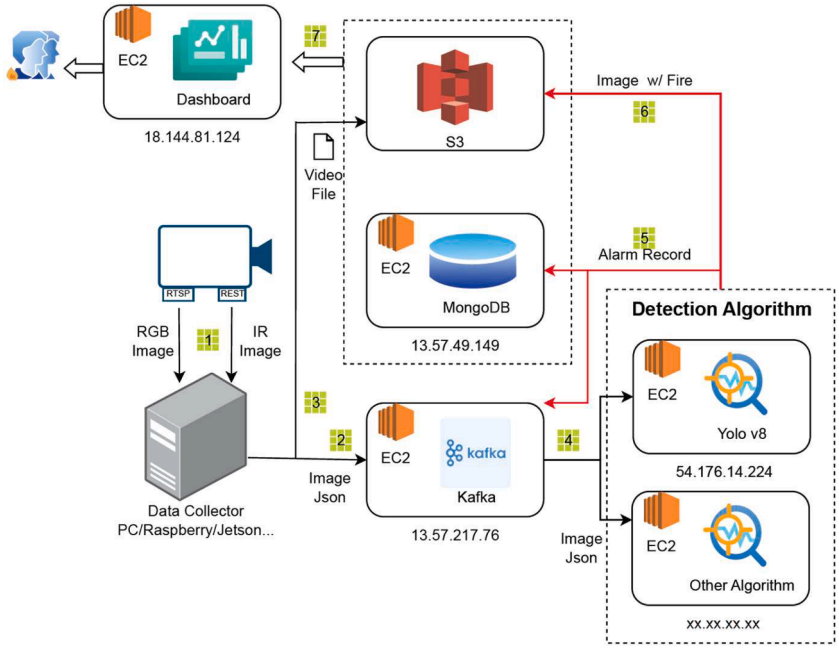


Fig. 8. Architecture of edge-cloud fire detection.

3.3. Edge computing architecture

To reduce latency in detecting fire and smoke, we can offload part of the processing tasks to edge devices located closer to the cameras. Initially, the analysis was performed in the cloud, leveraging its superior computing resources but potentially introducing additional network latency. By shifting these tasks from the cloud to edge devices, we can minimize resource usage and enhance user experience by decreasing the time it takes to detect and respond to incidents. This approach not only improves efficiency but also ensures quicker detection and response times, critical for real-time applications like fire and smoke detection. At the edge side, there is no message queue included. Instead, it performs detection directly and calls Discord Webhook to send messages to the Discord channel. The message includes alarm type, camera ID and capture time. By utilizing the computing resource on the edge device, it can reduce response time and get better flexibility.

3.4. Fire detection computing methods

In the fire detection system proposed in this study, multiple fire detection methods can be integrated into the system and they can simultaneously detect fire by consuming images from Kafka. In this research, three methods are researched, namely RGB image-based detection, IR image-based detection and RGB-IR enhanced image-based detection.

3.4.1. RGB image based detection

In this project, computer vision-based fire detection algorithms are applied to detect fire from images. Specifically, YOLO v8 is utilized in FDS.

YOLOv8 is an object detection model belonging to the You Only Look Once (YOLO) algorithm family. It leverages advanced neural network architectures and optimized algorithms to deliver high-performance object detection in various applications, such as surveillance and robotics [47]. The model's design focuses on balancing precision and recall, making it suitable for deployment in real-time systems where rapid and reliable object detection is crucial. The data used to train this fire detection model is provided in this GitHub project [48]. The trained weights are used to detect fire from the images collected from the FLIR thermal camera. Deploying the FDS with YOLOv8 on an AWS EC2 t4g.nano instance (1 vCPU, 0.5 GB RAM, 5 Gbps Bandwidth) results in an average fire detection time of 86 milliseconds per image. With a network bandwidth of 1 Gbps and an average image size of 100 Kb, the transmission time per image is approximately 0.8 milliseconds, which is negligible compared to the detection time using YOLOv8. When fire is detected by the YOLOv8 model, the fire alarm in JSON format will be sent to Kafka. An example sent to Kafka is shown in the below code snippets.

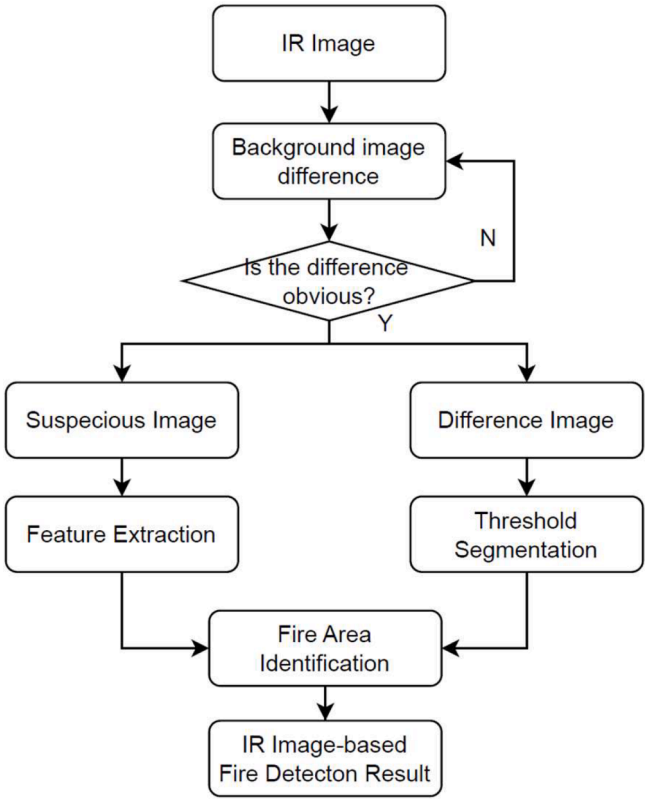


Fig. 9. Fire detection based on threshold segmentation method.

```
{
  "_id": {
    " $oid ": " 6674 ac128b6bec103f9144dd "
  },
  " camera id": "FLIR -A700 -85903894 ",
  " capture time ": {
    " $date ": " 2024 -06 -20 T22 :24:16.897 Z"
  },
  " alarm type ": " SMOKE ",
  " detect time ": {
    " $date ": " 2024 -06 -20 T22 :24:17.613 Z"
  },
  " detect model ": " YOLOv8 ",
  " bbox list ": {
    " probility ": 70.51,
    "x1": 86,
    "y1": 129,
    "X2": 867,
    "y2": 441
  },
  "s3 url": " https :// fire - clips .s3.us -west -1. amazonaws.com/FLIR -A700 -85903894 1718922256897. jpg"
}
```


3.4.2. Infrared image based detection

Wildfire flames exhibit characteristics such as flickering, layered changes, and overall movement, which are typically distinct from those of other high-temperature objects in infrared video. These features make it relatively easy to separate the wildfire flame area from the background. These characteristics are the primary basis for determining the occurrence of a wildfire and help to reduce false alarms and false reports, thereby further improving the accuracy of wildfire monitoring. The specific steps are described as follows shown in Fig. 9.

Step 1 - Use the background subtraction method to process the input infrared video to identify situations where there are no abnormalities. If an abnormality appears in the infrared video image, the abnormal frame and the difference image between the background and the abnormal frame need to be extracted, and steps 2 to 4 should be continued. If there are no abnormalities, the camera will continue to collect infrared video data and return to Step 1 to start monitoring again.
Step 2 - Compare the background image with the different images of the current frame and use the threshold segmentation method for processing. Threshold segmentation is an image processing technique that separates different regions in the image by setting a threshold.
Step 3 - Analyze the abnormal frames of the current infrared video image. Then, use a simple statistical method to perform threshold segmentation on the abnormal frame image to obtain the result of the abnormal frame segmentation.
Step 4 - During threshold segmentation, the initial threshold z_1 needs to be coarsely segmented first as shown in Eq. (1) and Eq. (2).

$$z_1 = \frac{\sum_x \sum_y n_{xy} f(x,y)}{\sum_x \sum_y n_{xy}} \tag{1}$$

$$n_{xy} = \max\left(|n_x|, |n_y|\right), n_y = f(x, y-1) - f(x, y+1) \tag{2}$$

where f(x, y) is the pixel value of the image. After calculating the initial threshold z_1, the optimal segmentation threshold z can be calculated using the maximum inter-class variance method. The maximum inter-class variance method segmentation aims to obtain the maximum variance between the image background and the feature target area. The formula for calculating the maximum variance is as shown in Eq. (3):

$$t_0 = \sum_{i=z}^{g} t_i, j_0 = \sum_{i=z}^{g} \frac{it_i}{t_0}, t_1 = \sum_{i=g+1}^{L} t_i, j_1 = \sum_{i=g+1}^{L} \frac{it_i}{t_1} \tag{3}$$

where, t_i represents the pixel value of the image, z is the initial threshold, g is the optimal segmentation threshold, L is the total number of gray levels in the image, t_0 is the sum of the pixel values from the initial threshold z to the optimal threshold g, j_0 is the weighted sum of these pixel values, t_1 is the sum of the pixel values from the optimal threshold $g+1$ to the highest gray level L, and

j_1 is the weighted sum of these pixel values. Based on the pixel values and means of the background and foreground, the inter-class variance $\sigma_B^2(g)$ at the current threshold g is calculated as Eq. (4):

$$\sigma_B^2(g) = w_0 \cdot w_1 \cdot (j_0 - j_1)^2 \tag{4}$$

where w_0 and w_1 are the weights of the background and foreground, respectively, defined as Eq. (5):

$$w_0 = \frac{t_0}{N}, w_1 = \frac{t_1}{N} \tag{5}$$

where N is the total number of pixels. The optimal threshold g is selected by iterating through all possible thresholds and choosing the one that maximizes the inter-class variance $\sigma_B^2(g)$. The larger the $\sigma_B^2(g)$, the more distinct the difference between the foreground (flame area) and the background, leading to better segmentation results. This helps to effectively isolate the flame area, thereby improving the accuracy of fire detection.

Compare and analyze the threshold segmentation image of the difference image obtained in step 2 with the threshold segmentation image of the abnormal frame obtained in step 3, in order to determine the suspected fire area.

Step 5 - Perform noise removal processing on the feature image of the suspected fire area, and outline the fire area with a red bright line box.

The wildfire detected based on the infrared image is marked with red boundaries as shown in Fig. 10.

3.4.3. RGB-Infrared image based detection

To utilize the capability of FLIR cameras to simultaneously capture RGB and infrared images, RGB-Infrared enhanced images can be used for detection. This helps to reduce false alarm rates and improve detection accuracy. The linear fusion algorithm is an effective image fusion method used to combine visible and infrared images. It achieves image fusion by applying linear weighting of the pixel values between the two images. This approach retains the main features of both types of images while preserving their individual characteristics in the fused image. The equation representing the linear fusion equation for infrared and visible light images is shown in Eq.(6).

$$F(x,y) = \alpha V(x,y) + \beta I(x,y) \tag{6}$$

This study continues to use the YOLO v8 model to detect images enhanced with different proportions, and the results are shown in Table 1.

In Table 1, through simulations, the infrared image is overlaid on the RGB image using a linear weighting method. By gradually increasing the weight of the IR image, it can be observed that when $\alpha = 0.9$ and $\beta = 0.1$, the likelihood of detecting flames and smoke increases. This indicates that the enhanced image can improve detection sensitivity to a certain extent. However, it is important to note that since the main focus of this study is to build an automated fire detection system, the enhanced image detection method still uses the YOLO v8 model. If higher detection capability is needed, the enhanced images should be used to retrain the model parameters, which I believe will result in improved detection performance.

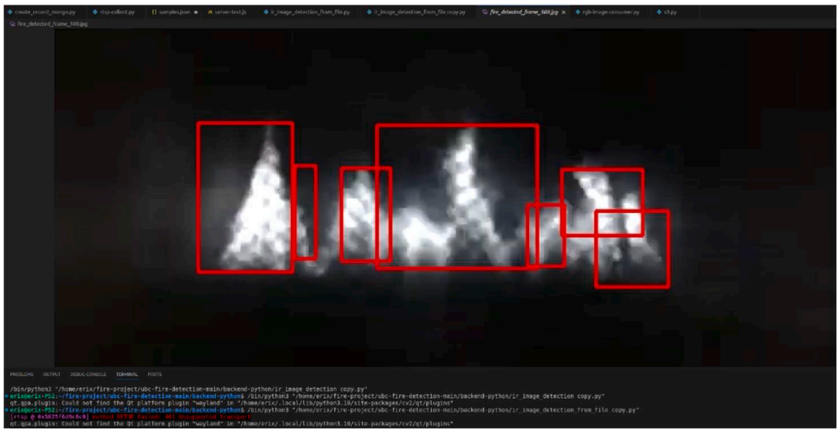


Fig. 10. Fire detection based on threshold segmentation method.

Table 1
Detection results.

Fire %	Smoke %	α	β
88.40	87.59	0.9	0.1
89.71	87.69	0.8	0.2
86.79	83.66	0.7	0.3
82.9	76.5	0.6	0.4
79.3	56.59	0.5	0.5
79.3	56.59	0.4	0.6
72.58	73.59	0.3	0.7

4. Proof of concept UBC Okanagan Campus building fire safety

4.1. Deployment

In the proof of concept experiments, FLIR A700 which is shown in Fig. 11, was used as the thermal camera. The detailed specification information is listed in Table 2. This camera provides an RTSP stream for RGB data and thermal data. The RTSP streaming URL is in Table 3. FLIR supports streaming a variety of video formats, including MPG4, MJPG and h.264. In my experiment, we choose h.264 to run all tests. The video configuration options are listed in Table 4.

All the background services and middleware are deployed on AWS EC2 servers, shown in Table 5. Since the FLIR camera is not assigned with a public address, we can only deploy the DCS on a server in the same local area network. All the background services can be deployed with more instances to meet the requirement as the number of cameras increases and the scalability is considered in the design. On the edge side, this study deploys image data collection and fire detection tasks based on the Raspberry Pi 4B, aiming to significantly reduce latency and improve detection reliability. The Raspberry Pi 4B is a powerful small computer equipped with GPU support for OpenGL ES 3.x, hardware-accelerated OpenVG, and up to 4 K HEVC video hardware decoding capabilities, making it highly suitable for processing image data and performing real-time fire detection tasks. Thanks to the excellent performance of the Raspberry Pi 4B, it is utilized as an edge device capable of capturing one image per second from the video stream. Using YOLOv8, a detection result can be obtained in 80 milliseconds, and the alert information can be pushed to a Discord group via a webhook within one second, allowing group members to view fire or smoke alerts generated by the system.

4.2. Dashboard DEMOS

On the cloud side, detection results and image bytes are pushed to a Kafka topic. The FDS program subscribes to this Kafka topic, uploads the image bytes to AWS S3, and retrieves an image URL for display purposes. This program also publishes the detection results into MongoDB for future queries. Using a message queue like Kafka helps manage the flow of requests and prevents direct overwhelming access to the database. If too many requests hit the database simultaneously, it could crash. Also, since the Kafka message queue system is deployed on AWS EC2, any authorized research team that is interested in conducting research using the image collected by our system can build and test their new fire detection methods. To reduce latency in detecting fire and smoke, we can offload part of the processing tasks to edge devices located closer to the cameras. Initially, the analysis was performed in the cloud, leveraging its superior computing resources but potentially introducing additional network latency. By shifting these tasks from the cloud to edge devices, we can minimize resource usage and enhance user experience by decreasing the time it takes to detect and respond to incidents. This approach not only improves efficiency but also ensures quicker detection and response times, critical for real-time applications like fire and smoke detection. On the edge side, it does not include a message queue. Instead, it performs detection directly and calls Discord webhook to send messages to the discord channel. The message includes alarm type, camera ID and capture time. The dashboard of the system is shown in Fig. 12. In this study, Amazon Web Services are used to deploy the cloud-based services and Raspberry Pi4 is used to deploy edge services.

5. Discussion

This study proposes an automated fire detection and alarm system that utilizes thermal cameras to collect RGB and infrared images and detects fire with edge computing and a cloud-based platform. The design of this system showcases several strengths when compared to the existing literature. Firstly, the system can capture images in real-time from the camera's RTSP stream and transmit them through Kafka, ensuring the timeliness of fire detection. Secondly, storing data on the AWS cloud not only provides a large-scale data storage capacity but also facilitates data backup and recovery. Thirdly, the design of the DCS allows for the addition of more instances as needed to support the integration of more cameras, enhancing the system's scalability. Lastly, this system has the flexibility and immediate update of the algorithms, which means the algorithms running on the cloud can be updated and compared easily, and the fire detection algorithms can be continuously optimized to improve accuracy. This is significant for enhancing the fire resilience in urban areas. In terms of robustness, each camera is dedicated solely to image capture and transmission, while fire detection is performed by the FDS in the cloud. By scaling cloud-based services, the system avoids single points of failure and ensures that detection capacity matches the volume of images being processed. For cameras monitoring critical areas, edge devices are deployed for local image capture and detection, providing redundancy. This ensures fire detection remains operational even in the

Fig. 11. FLIR A700 thermal camera.

Table 2
Camera specifications.

Parameter	Value
Thermal Resolution	640 × 480
Visual Resolution	1280 × 960
Lens	14°, 42°
Image Frequency	30Hz
Object Temperature Range	−20 °C to 2000 °C

Table 3
RTSP url for RGB and thermal video stream in h.264 format.

Type	Video Type	RTSP Streaming URL
RGB	H.264	rtsp://169.254.117.169/avc/ch1
Thermal	H.264	rtsp://169.254.117.169/avc?ch0

Table 4
Streaming configuration provided by FLIP A700.

Attribute	Description
Frate	frame rate in Hz
Overlay	on/off
Cbr	constant bitrate in bits per second, 0=variable bit rate
Quant	quality level −1 to 100 where 100 is best quality, valid only if cbr=0
Gop	group of pictures, gop is the sum of one key frame + predicted frames, in other words the number of P-frames following an I-frame

Table 5
System deployment plan.

Instance Name	Instance No.	Description
MongoDB	1	MongoDB deployed using Docker
Kafka	2	Zookeeper and Kafka deployed using Docker
FDS	3	FDS with YOLO v8, programmed in Python 3.10
Dashboard	4	Dashboard providing graphical user interface, programmed in React

event of network disruptions, maintaining system availability.

The proposed design can be further improved in the following areas. For example, the system relies on cloud services and a large amount of hardware equipment, which may incur high costs, especially when deployed on a large scale, as the fees from cloud service providers will increase with usage. Secondly, the performance of the system largely depends on the stability and performance of third-party services such as AWS. As the image data is transmitted through the message queue Kafka, and the images and videos where a fire is detected will be stored on AWS, this may raise concerns about privacy and data protection.

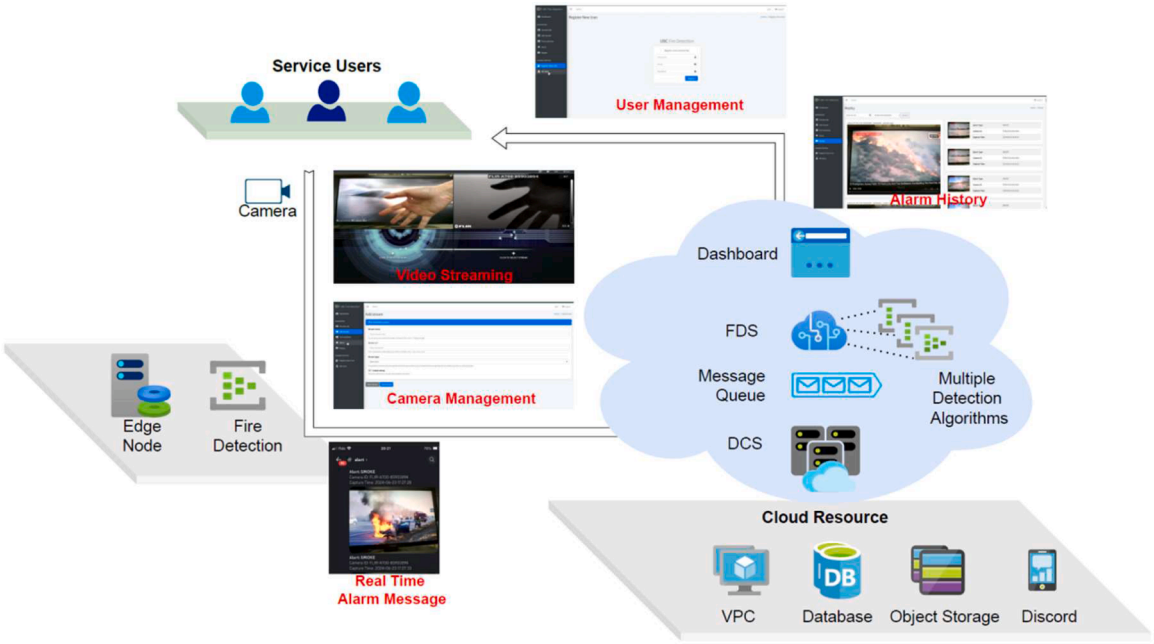


Fig. 12. Dashboard demo.

6. Conclusions and future work

To cope with wildfire threats, this study focused on the development of an automated fire detection and alarm system by leveraging edge computing and cloud-based platforms, specifically targeting urban local area fire detection. By integrating FLIR thermal cameras, the system captures both visible and infrared images and detects fire in multiple algorithms, significantly enhancing overall detection reliability. The use of computer vision techniques, including both visible and infrared imaging, allows for accurate detection of fire characteristics such as flames and smoke, even in complex environments. Implementing edge computing reduces latency in fire detection and alarm activation to enable faster responses . Additionally, the cloud platform facilitates data storage, analysis, and remote monitoring to ensure scalability and accessibility of fire detection data.

Future research will focus on optimizing the detection algorithms to further reduce false alarms and improve detection accuracy in various environmental conditions. Enhancing sensor fusion techniques to seamlessly integrate data from multiple sensors would provide a more comprehensive fire detection system. Incorporating advanced AI and machine learning models could enable the prediction of potential fire hazards and improve the system's adaptability to different scenarios. Extensive field testing and multiple case studies using the proposed system in diverse urban environments will be required in future work to validate the system's robustness and reliability.

CRediT authorship contribution statement

Xueliang Yang: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Yenchun Li:** Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Qian Chen:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The case study of automated fire detection at the UBC Okanagan Campus reported in this research was supported by the Campus As A Living Lab Program Grant from the Office of the Vice-President Research and Innovation at the University of British Columbia (Grant No. AWD-027461 UBCVPFIO 2023).

Appendix A. MongoDB Data Structure

A.1. Fire Detection Alarm

Collection Name: RGB_ALARM_RECORD/IR_ALARM_RECORD
Data Model: Table 6

Table 6
Alarm data model in MongoDB.

Field Name	Data Type	Description
camera_id	String	The unique identifier for the camera that captured the data.
capture_time	String	The timestamp when the data was captured by the camera.
alarm_type	String	The type of alarm triggered: FIRE, SMOKE, or NONE.
detect_time	String	The timestamp when the detection process was completed.
detect_model	String	The model used for detection: YOLO v8, FIRENET, or RES50.
url	String	The URL of the S3 video footage.
bbox_list	Array of Objects	A list of bounding boxes representing the positions of detected fires. Each object contains the following fields: *probability:* The probability of fire, unit: %. *x1:* The x-coordinate of the upper-left corner of the bounding box. *y1:* The y-coordinate of the upper-left corner of the bounding box. *x2:* The x-coordinate of the lower-right corner of the bounding box. *y2:* The y-coordinate of the lower-right corner of the bounding box.

A.2. Camera information

Collection Name: CAMERA_INFO
Data Model: Table 7

Table 7
Camera information data model.

Field Name	Data Type	Description
camera_id	String	The unique identifier for the camera that captured the data.
rtsp_url	String	The RTSP stream address of the camera.
created_date	String	The timestamp when the camera information created.

Appendix B. Kafka data structure

B.1. Image captured

Topic Name: IMAGE_CAPTURE (for RGB Images), IR_IMAGE_CAPTURE (for IR Images)
Data Model: Table 8

Table 8
Captured image in Kafka.

Field Name	Data Type	Description
camera_id	String	The unique identifier for the camera that captured the data.
image_bytes	String	The image data in hexadecimal format.
capture_time	String	The timestamp when the image was captured, formatted as an ISO 8601 string.

B.2. Alarm produced by FDS

Topic Name: RGB_ALARM (for RGB Images), IR_ALARM (for IR Images)
Data Model: Table 9

Table 9
Alarm data model in Kafka.

Field Name	Data Type	Description
camera_id	String	The unique identifier for the camera that captured the data.
capture_time	String	The timestamp when the data was captured by the camera.
alarm_type	String	The type of alarm triggered: FIRE, SMOKE, or NONE.
detect_time	String	The timestamp when the detection process was completed.
detect_model	String	The model used for detection: YOLO v8, FIRENET, or RES50.
url	String	The URL of the S3 video footage.
bbox_list	Array of Objects	A list of bounding boxes representing the positions of detected fires. Each object contains the following fields: *probability:* The probability of fire, unit: %. *x1:* The x-coordinate of the upper-left corner of the bounding box. *y1:* The y-coordinate of the upper-left corner of the bounding box. *x2:* The x-coordinate of the lower-right corner of the bounding box. *y2:* The y-coordinate of the lower-right corner of the bounding box.

Data Availability Statements

Some or all data, models, or simulation codes that support the findings of this study are available from the corresponding author upon reasonable request.

Data availability

Data will be made available on request.

References

[1] S. Sankey, Technical Coordinator, Blueprint for Wildland Fire Science in Canada (2019–2029), Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre, Edmonton, AB, 2018.
[2] M.R. Kreider, P.E. Higuera, S.A. Parks, W.L. Rice, N. White, A.J. Larson, Fire suppression makes wildfires more severe and accentuates impacts of climate change and fuel accumulation, Nat. Commun. 15 (1) (2024) 2412.
[3] UBCM, Statistics confirm devastating 2023 wildfire season, https://www.ubcm.ca/about-ubcm/latest-news/statistics-confirm-devastating-2023-wildfire-season, December 20, 2023.
[4] Copernicus Atmosphere Monitoring Service, 2023: a year of intense global wildfire activity, https://atmosphere.copernicus.eu/2023-year-intense-global-wildfire-activity, January 15, 2024.
[5] E. De Luigi, New German technology aims to quickly detect wildfires, Firefighting in Canada, https://www.firefightingincanada.com/new-german-technology-aims-to-quickly-detect-wildfires/, June 23, 2023.
[6] Q. Chen, B. Garcia de Soto, B.T. Adey, Construction automation: Research areas, industry concerns and suggestions for advancement, Autom. Constr. 94 (2018) 22–38.
[7] Y. Cui, H. Dong, E. Zhou, An early fire detection method based on smoke texture analysis and discrimination, in: Congress on Image and Signal Processing (CISP), IEEE, 2020.
[8] K. Dimitropoulos, P. Barmpoutis, N. Grammalidis, Spatio-temporal flame modeling and dynamic texture analysis for automatic video-based fire detection, IEEE Trans. Circuits Syst. Video Technol. 25 (2) (2020) 339–351.
[9] Z. Xu, Y. Guo, J.H. Saleh, Advances toward the next generation fire detection: deep LSTM variational autoencoder for improved sensitivity and reliability, IEEE Access 9 (2021) 30636–30653.
[10] A.S. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P.Z. Fulé, E. Blasch, Aerial imagery pile burn detection using deep learning: the flame dataset, Comput. Netw. 193 (2021).
[11] F. Khan, Z. Xu, J. Sun, F.M. Khan, A. Ahmed, Y. Zhao, Recent advances in sensors for fire detection, Sensors 22 (9) (2022) 3310.
[12] D.T. Gottuk, M.J. Peatross, R.J. Roby, C.L Beyler, Advanced fire detection using multi-signature alarm algorithms, Fire Saf. J 37 (4) (2002) 381–394.
[13] E. Zervas, A. Mpimpoudis, C. Anagnostopoulos, O. Sekkas, S. Hadjiefthymiades, Multisensor data fusion for fire detection, Inf. Fusion 12 (3) (2011) 150–159.
[14] R.W. Bukowski, R.D. Peacock, J.D. Averill, T.G. Cleary, N.P. Bryner, P.A. Reneke, Performance of Home Smoke alarms: Analysis of the Response of Several Available Technologies in Residential Fire Settings, NIST Technical Note 1455-1 (2003).
[15] G. Xiao, H. Weng, L. Ge, Q. Huang, Application status of carbon nanotubes in fire detection sensors, Front. Mater. 7 (2020) 588521.
[16] Q. Su, G. Hu, Z. Liu, Research on fire detection method of complex space based on multi-sensor data fusion, Meas. Sci. Technol. 35 (8) (2024) 085107.
[17] B.U. Töreyin, Y. Dedeoğlu, U. Güdükbay, A.E. Cetin, Computer vision based method for real-time fire and flame detection, Pattern Recognit. Lett. 27 (1) (2006) 49–58.
[18] T.H. Chen, Y.H. Yin, S.F. Huang, Y.T. Ye, The smoke detection for early fire-alarming system base on video processing, in: *2006 international conference on intelligent information hiding and multimedia*, IEEE, 2006, pp. 427–430.
[19] J. Sharma, O.C. Granmo, M. Goodwin, J.T. Fidje, Deep convolutional neural networks for fire detection in images, Eng. Appl. Artif. Intell. 105 (2021).
[20] Y. Liu, X. Shi, Y. Ren, and L. Jiao, Computer vision-based fire detection methods: a survey, Knowledge-Based Systems, vol. 250, pp. 109081, 2022.
[21] S. Fouziya Sulthana, C.T. Asha Wise, C.V. Ravikumar, R. Anbazhagan, G. Idayachandran, G. Pau, Review study on recent developments in fire sensing methods, IEEE Access 11 (2023) 90269–90282.
[22] S. Sharma, O.C. Granmo, M. Goodwin, J.T. Fidje, Deep convolutional neural networks for fire detection in images, Eng. Appl. Artif. Intell. 105 (2021) 104422.
[23] Y. Liu, X. Shi, Y. Ren, L. Jiao, Computer vision-based fire detection methods: a survey, Knowl Based Syst 250 (2022) 109081.
[24] P. Jain, S.C. Coogan, S.G. Subramanian, M. Crowley, S. Taylor, M.D. Flannigan, A review of machine learning applications in wildfire science and management, Environ. Rev. 28 (4) (2020) 478–505.
[25] S. Wu, X. Zhang, R. Liu, B. Li, A dataset for fire and smoke object detection, Multimed Tools Appl 82 (2023) 6707–6726.
[26] C. Jin, Video Fire Detection Methods Based on Deep Learning: datasets, Methods, and Future Directions, Fire 6 (2023) 315.

[27] A. Khan, B. Hassan, S. Khan, R. Ahmed, A. Abuassba, DeepFire: a Novel Dataset and Deep Transfer Learning Benchmark for Forest Fire Detection, Mobile Information Systems 2022 (2022) 5358359. Article ID.
[28] K. Jiang, T. Xie, R. Yan, X. Wen, D. Li, H. Jiang, N. Jiang, L. Feng, X. Duan, J. Wang, An attention mechanism-improved YOLOv7 object detection algorithm for hemp duck count estimation, Agriculture 12 (2022) 1659.
[29] D. Gragnaniello, A. Greco, C. Sansone, B. Vento, Fire and smoke detection from videos: a literature review under a novel taxonomy, Expert Syst. Appl. 255 (2024) 124783.
[30] C.S. Qin, M.L. Zhang, W. He, et al., A new real-time fire detection method based on infrared image, in: 7th IEEE International Conference on Computer Science and Network Technology (ICCSNT), 2019, pp. 476–479.
[31] C. Yuan, Z.X. Liu, Y.M. Zhang, et al., Fire detection using infrared images for UAV-based forest fire surveillance, in: International Conference on Unmanned Aircraft Systems (ICUAS), 2017, pp. 567–572.
[32] W.H. Kim, S.K. Kim, J.H. Lee, et al., A fire alarm vision system based on IR image processing, in: The 5th International Conference on New Trends in Information Science and Service Science 2, 2011, pp. 291–293.
[33] H. Shekhar, S.J. Kumar, P.S. Rajawat, Multi infrared (IR) flame detector for tangential fired boiler, in: V.V. Das, Y. Chaba (Eds.), Mobile Communication and Power Engineering, Springer, Berlin, 2013, pp. 545–548.
[34] B.C. Arrue, A. Ollero, J.R.M. de Dios, An intelligent system for false alarm reduction in infrared forest-fire detection, IEEE Intellig. Syst. their Appl. 15 (3) (2000) 64–73.
[35] Y.B. Wang, X.M. Ma, Early fire detection for high space based on video-image processing, in: International Symposium on Computer, Consumer and Control (IS3C), 2014, pp. 785–788.
[36] Y. Li, A. Zou, L. Ye, Multi-feature fusion fire detection algorithm based on faster R-CNN, Fire Technol. 57 (2021) 451–471.
[37] M.M. Aslam, K. Riaz, S.A. Hussain, N.K. Mohsin, Convolutional neural network-based fire detection in surveillance videos, J. Intellig. Syst. 30 (1) (2021) 624–634.
[38] K.W. Wang, Y.M. Zhang, J.J. Wang, Q. Zhang, B. Chen, D. Liu, Fire detection in infrared video surveillance based on convolutional neural network and SVM, in: IEEE 3rd International Conference on Signal and Image Processing (ICSIP), 2018, pp. 162–167.
[39] J. Li, Z. Li, O. Koyejo, J. Li, Automatic fire detection in infrared images for unmanned aerial vehicles-based forest fire surveillance, Int. J. Distrib. Sens. Netw. 16 (6) (2020).
[40] Y. Zhang, X. Wan, H. Shi, L. Pan, Research and application of fire detection algorithm based on convolutional neural network in infrared image, Infrared Phys. Technol. 109 (2020) 103423.
[41] S. Akhloufi, A. Ben Mhamed, R. Audet, Deep learning-based approach for wildland fire detection in UAV imagery, Remote Sens (Basel) 13 (14) (2021).
[42] Z. Liu, G. Qiu, Y. Gao, Wildfire detection from infrared images using deep learning and handcrafted features, Fire Technol. 57 (2021) 1885–1913.
[43] S. Modi, Y. Lin, L. Cheng, G. Yang, L. Liu, W.J. Zhang, A socially inspired framework for human state inference using expert opinion integration, IEEE/ASME Trans. Mechatron. 16 (5) (2011) 874–878. Oct.
[44] W. Zhang, J. Wang, Y. Lin, Integrated design and operation management for enterprise systems, Enterprise Inf Syst 13 (4) (2019) 424–429.
[45] Z. Bi, Y. Lin, W.J. Zhang, The general architecture of adaptive robotic systems for manufacturing applications, Robot Comput Integr Manuf 26 (5) (2010) 461–470.
[46] A. Arasu, M.G. Katz, A.K. Singh, Consistent Hashing for Load Balancing in Distributed Systems, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1367–1380.
[47] X. Chen, J. Wu, Z. Liu, X. Xu, YOLOv8: a New Architecture for Real-Time Object Detection and Segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 1254–1264.
[48] Y. Geng, fire-smoke-detect-yolov4, GitHub repository, 2024. [Online]. Available: https://github.com/gengyanlei/fire-smoke-detect-yolov4.