SafeZone - Real-time Video Analytics for Industrial Safety

1 INTRODUCTION

1.1 Overview

SafeZone is a cutting-edge project that employs real-time video analytics to bolster industrial safety in diverse industrial environments. The project's primary objective is to monitor and swiftly identify potential safety hazards, including unauthorized access to restricted areas, equipment malfunctions, and non-compliance with safety protocols. By harnessing advanced computer vision techniques and machine learning algorithms, SafeZone scrutinizes live video feeds from strategically placed surveillance cameras within the industrial facility. The system adeptly recognizes objects, activities, and behaviors, providing real-time alerts to prevent accidents and enhance overall workplace safety. SafeZone is an invaluable tool that proactively safeguards workers and assets in industrial settings.

1.2 Purpose

The primary purpose of the SafeZone project is to create a proactive safety system that effectively identifies and mitigates potential hazards within industrial settings. Through real-time monitoring and immediate alerts, this project aims to prevent accidents, unauthorized access, equipment malfunctions, and non-compliance with safety protocols. By harnessing the power of computer vision and AI-driven algorithms, SafeZone strives to provide a safer working environment for employees while safeguarding valuable assets.

2 LITERATURE SURVEY

2.1 Existing Problem

Industrial safety has been a persistent challenge, with traditional safety measures sometimes falling short in preventing accidents and hazards. Conventional methods often rely on manual inspection, which can be time-consuming, prone to errors, and less effective in rapidly evolving situations.

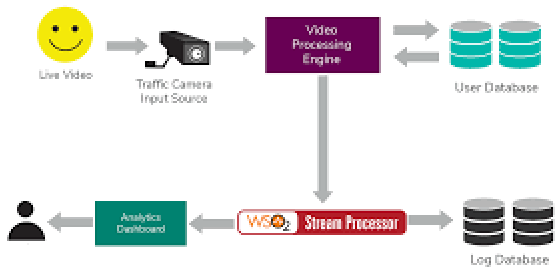
2.2 Proposed Solution

The SafeZone project proposes a solution that leverages real-time video analytics, machine learning, and AI technologies. By continuously monitoring live video feeds from strategically placed cameras, the system aims to provide swift and accurate detection of potential safety hazards. This approach not only improves hazard identification but also facilitates prompt intervention and prevention.

3 THEORITICAL ANALYSIS

3.1 Block diagram

Video data is grabbed from the CCTV cameras. The video processing engine processes the video frames, detects the available faces, and cross-checks the features with the available user database. Then the extracted data is sent to WSO2 Stream Processor to be processed. User movement and behaviors are logged into the log database and real-time analysis results are forwarded to the user via analytics dashboard.



3.2 Hardware / Software designing

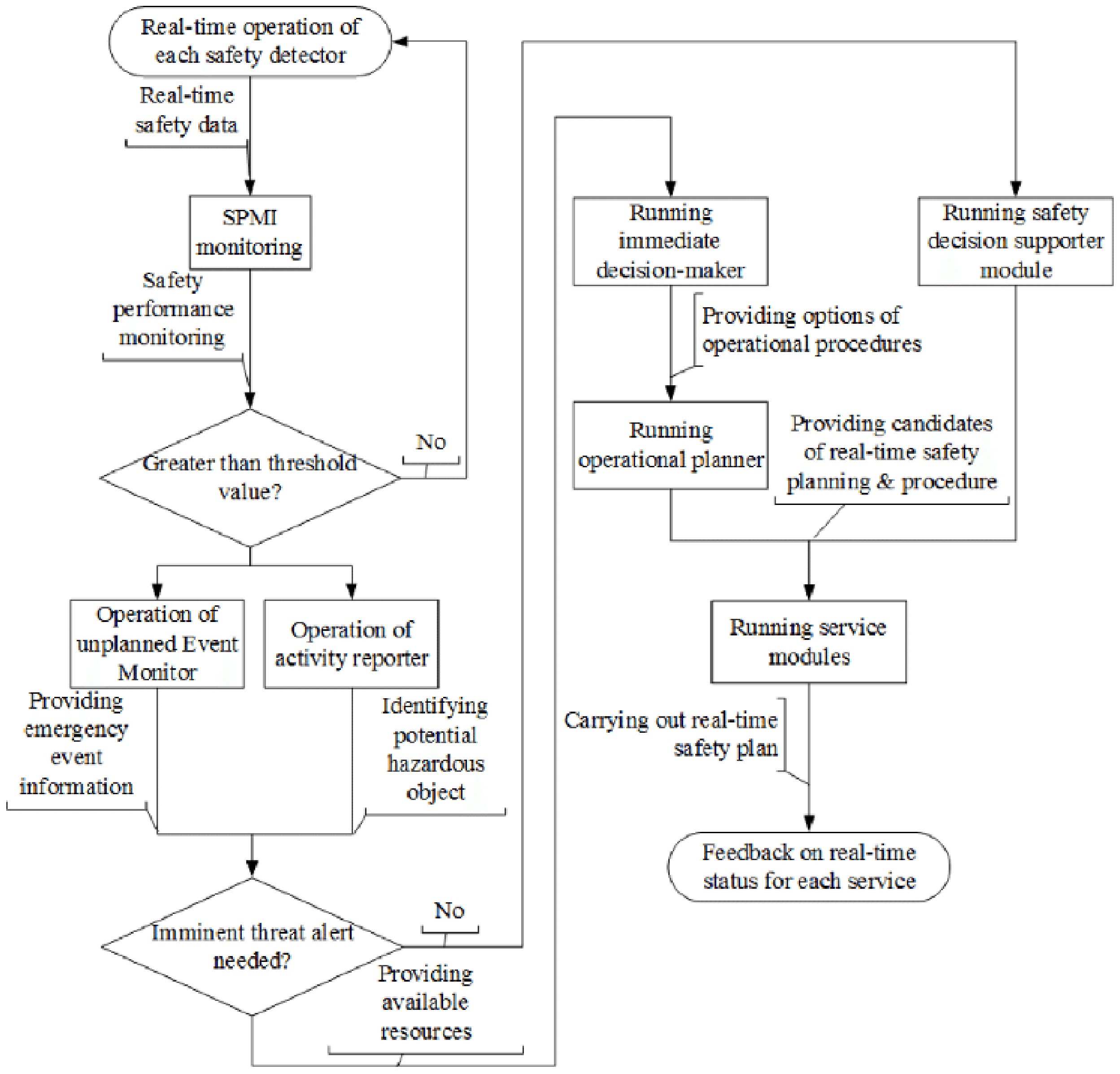
The hardware requirements include strategically positioned surveillance cameras capable of capturing high-quality video feeds. The software component involves proficiency in Python for implementing machine learning algorithms, as well as libraries such as Numpy and Pandas for data manipulation. The development of a user interface using Python-Flask is also a crucial part of the project.

4 EXPERIMENTAL INVESTIGATIONS

This section delves into the practical aspects of the project:

- Data Collection: Describe how video data from industrial environments was collected and prepared for analysis.
- Algorithm Training: Detail the process of training machine learning models to recognize objects, activities, and behaviors indicative of safety hazards.
- Real-Time Monitoring: Explain how the system was implemented to process live video feeds in real time and generate alerts when potential hazards are identified.
- Performance Evaluation: Provide insights into the accuracy and efficiency of the system in identifying hazards compared to manual methods.
- Case Studies: Share specific instances where the system successfully identified and prevented potential safety hazards, showcasing its effectiveness.

5 FLOWCHART



Real-time operation of each safety detector
Real-time safety data
SPMI monitoring
Safety performance monitoring
Greater than threshold value?
No
Operation of unplanned Event Monitor
Operation of activity reporter
Providing emergency event information
Identifying potential hazardous object
Imminent threat alert needed?
No
Providing available resources
Running immediate decision-maker
Running safety decision supporter module
Providing options of operational procedures
Running operational planner
Providing candidates of real-time safety planning & procedure
Running service modules
Carrying out real-time safety plan
Feedback on real-time status for each service

6 ADVANTAGES & DISADVANTAGES

Advantages:

- Proactive Safety: Real-time alerts allow for immediate intervention, reducing accidents.
- Accurate Hazard Detection: Machine learning models improve accuracy in identifying hazards.
- Continuous Monitoring: The system operates 24/7, ensuring round-the-clock safety.
- Improved Compliance: Ensures adherence to safety protocols and regulations.
- Enhanced Efficiency:Minimizes downtime by promptly detecting equipment malfunctions.

- User-Friendly Interface: Intuitive dashboard facilitates easy system interaction.
- Data-Driven Insights: Provides valuable data for optimizing safety measures.

Disadvantages:

- Initial Setup: Requires investment in hardware, cameras, and software development.
- Training Complexity: Developing accurate machine learning models demands expertise.
- False Positives: System may generate alerts for benign events, leading to alert fatigue.
- Maintenance: Cameras and hardware need regular maintenance and updates.

7 APPLICATIONS

- Manufacturing Plants: Enhance safety for workers and equipment on the factory floor.
- Construction Sites: Monitor construction activities and prevent accidents.
- Oil & Gas Industry: Improve safety in hazardous environments.
- Warehouses: Prevent unauthorized access and track movements.
- Mining Operations: Monitor activities in remote and risky locations.
- Transportation Hubs: Enhance security at airports, train stations, etc.
- Healthcare Facilities: Ensure safety protocols are followed.

8 CONCLUSIONS

In conclusion, the SafeZone project's real-time video analytics solution presents a cutting edge approach to industrial safety. By integrating computer vision, machine learning, and swift response mechanisms, the system proactively identifies hazards and enhances overall safety in industrial environments. This innovative solution empowers businesses to safeguard personnel, assets, and operations more effectively.

9 FUTURE SCOPE

- Multi-Sensor Integration: Combine video analytics with other sensors for comprehensive safety.
- Remote Monitoring: Develop mobile apps for real-time monitoring from anywhere.
- AI Refinement: Improve machine learning models for higher accuracy.
- Predictive Analytics: Forecast potential hazards using historical data.
- Cloud Integration: Store and analyze data in the cloud for scalability.
- Automated Actions: Integrate systems to respond to certain alerts automatically.

10 BIBILOGRAPHY

[1] V. Ganesan, Y. Ji, M. Patel "Video meets the Internet of Things". [Online].Available: https://www.mckinsey.com/industries/high-tech/our-insights/video-meets-the-internetof-things

[2] OpenCV.org,'About OpenCV',2016.[Online]. Available: https://opencv.org/about.html

[3] Tensorflow.org,'About Tensorflow',[Online].Available: https://www.tensorflow.org/

[4] ImageNet.org, 'Image Net',[Online].Available: https://www.image-net.org/

[5] A. Krizhevsky, I. Sutskever, G.E Hinton "ImageNet Classification with Deep Convolutional Neural Networks" in Neural Information Processing Systems Conference 2012.

[6] Henning Schulzrinne, RTSP:Real-Time Streaming Protocol (RTSP), https://www.cs.columbia.edu/~hgs/rtsp/

[7] WSO2.com,'WSO2 Stream Processor'.[Online]. Available: https://wso2.com/analytics/

[8] WSO2.com,'WSO2 Identity Server',[Online].Available: https://wso2.com/products/identity-server/

[9] Gama, J., & Brazdil, P. (2000). Cascade generalization. Machine Learning, 41(3), 315- 343

[10] Kuihe Yang, Zhiming Cai, Lingling Zhao (2013), Algorithm Research on Moving Object Detection of Surveillance Video Sequence, Optics and Photonics Journal, pp. 308-312.

[11] Github.com, OpenCV_PlaneDetection, [Online]. Available https://github.com/wso2-incubator/video-image-preprocessing-wso2.git

[12] Bovik, A.C. (2010), Handbook of Image and Video Processing, Elsevier Science, Pp.1282-1285

[13] Chris Harris and Mike Stephens (1988), A combined corner and edge detector, In Proc. of Fourth Alvey Vision Conference, pp.147-151.

[14] Lowe, D. G. (2004). Distinctive image features from scale-invariant key points. International journal of computer vision, 60(2) , 91-110.

[15] H. Chen, D. Chen and X. Wang, "Intrusion detection of specific area based on video," 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Datong, 2016, pp. 23-29.

APPENDIX

A. Source Code

```python
# Import necessary libraries
import cv2
import numpy as np

# Simulated function for hazard detection using machine learning model
def detect_hazard(frame):
    # Simulate detection based on a simple condition (e.g., red color)
    red_lower = np.array([0, 0, 100])
    red_upper = np.array([100, 100, 255])
    mask = cv2.inRange(frame, red_lower, red_upper)
    return mask

# Open a video capture from a file or webcam
cap = cv2.VideoCapture(0)  # Use 0 for webcam, or provide video file path

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Perform hazard detection using machine learning
    hazard_mask = detect_hazard(frame)

    # Display the original frame and the hazard mask
    cv2.imshow('Original Frame', frame)
    cv2.imshow('Hazard Detection', hazard_mask)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture and close windows
cap.release()
cv2.destroyAllWindows()

#The provided code simulates a real-time video feed and applies hazard
detection based on a color filter.
```

```
#In this simplified example, the hazard is detected using a simulated "red
color" condition.
```

When you run this code, you'll see two windows:

1. The "Original Frame" window displays the live video feed from your webcam.
2. The "Hazard Detection" window displays the result of hazard detection using the red color filter.

Capturing and Displaying Video:

To display video frames in Colab, you can create a loop that captures video frames and displays them using IPython.display:

```
import cv2
from IPython.display import display, clear_output
from google.colab.patches import cv2_imshow  # For displaying OpenCV
images

cap = cv2.VideoCapture(0)  # Use 0 for webcam

while True:
    ret, frame = cap.read()
    if not ret:
        break

    cv2_imshow(frame)  # Display OpenCV image

    clear_output(wait=True)  # Clear previous output

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

In this approach, the video frames are displayed using cv2_imshow() from the google.colab.patches module.

Capturing Output:

```
out = cv2.VideoWriter('output.avi', cv2.VideoWriter_fourcc(*'XVID'), 20.0,
(640, 480))  # Adjust dimensions

while True:
    ret, frame = cap.read()
    if not ret:
        break

    out.write(frame)  # Write frame to video

    clear_output(wait=True)  # Clear previous output

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

out.release()
cap.release()
cv2.destroyAllWindows()
```

This will save the captured frames as a video file named 'output.avi'.