Saras AI
CLASH OF T-AI-TANS

COMPUTER VISION HACKATHON

TEAM OF TITANS

Ashutosh Kumar
Biswajit Bera
Hirakjyoti Medhi
Roshan Jha

Problem Statement

There is a critical need for an automated surveillance solution that can continuously and accurately monitor environments for signs of emergencies, providing real-time alerts to facilitate rapid intervention.

Traditional systems relying on human operators are often slow and prone to errors, which can result in severe consequences, including loss of life and extensive property damage.

Our project aims to develop a real-time emergency surveillance system that leverages computer vision to detect and respond to critical situations such as fires, violence, and medical emergencies.

Proposed Solution

Part 1: **Violence Detection**

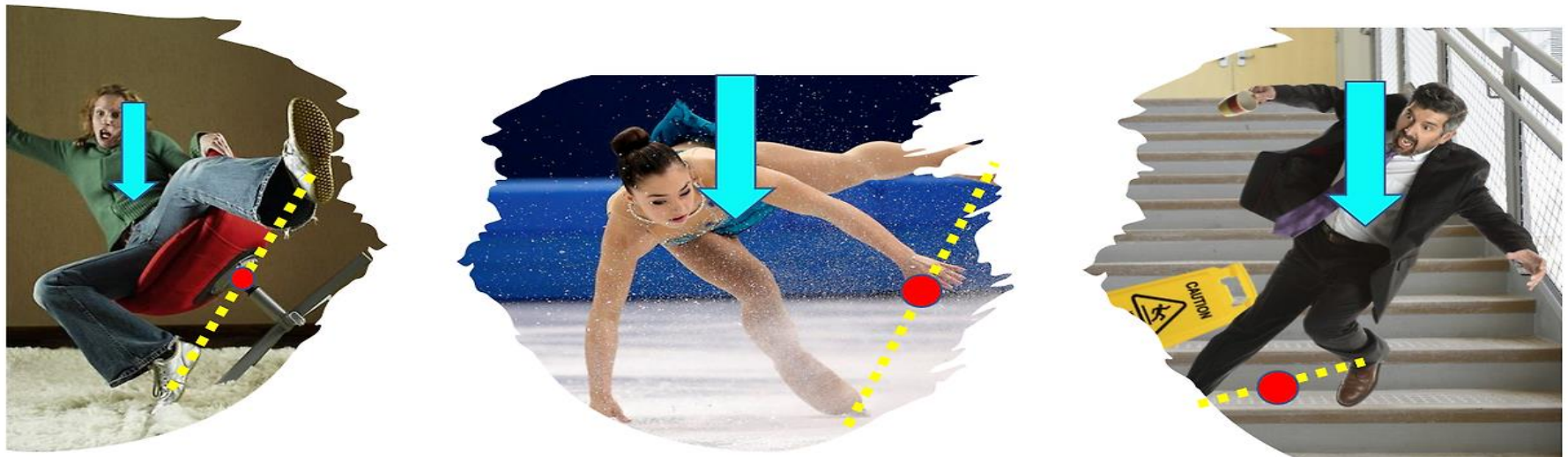
- Dataset: Utilized 1,000 videos each for violence and non-violence categories. Trained the MobileNetV2 model using this dataset.
- Model: It consists of a MobileNet pretrained model as a spatial feature extractor and Bi-LSTM as temporal relation learning method with a focus on the 3-factor generality-accuracy-quick response.
- **MobileNetV2**, a 53-layer deep convolutional neural network. Known for efficiency on resource-limited devices. Features an inverted residual structure with thin bottleneck layers. Employs lightweight convolutions in the expansion layer to reduce computational load.
- Functionality: Processes real-time video footage by capturing frames. Predicts violence probability from the captured frames.
- Evaluation: The suggested model achieved about 97% accuracy with speed of 16 frames/sec.

Part 2: **Fire Detection**

- Libraries: The system uses cv2 for image and video processing, threading for concurrent execution.
- Loading the Pre-Trained Model: The **cv2.CascadeClassifier** is used to load a pre-trained fire detection model from an XML file. The file contains data from a model trained on images with and without fire, allowing it to detect fire patterns in new images.
- How Cascade Classifier Works: The model processes video frames by scaling the image and sliding a window across different regions. Features like edges and textures are extracted from each window and compared against the patterns in the pre-trained model. The classifier uses a cascading process, quickly eliminating areas without fire and focusing on regions that potentially contain fire.
- Real-Time Detection: The system captures frames from a video feed, applies the trained model to detect fire, and triggers an alarm sound if fire is detected.

Part 3: **Fall Detection**

- Pose Estimation: Utilizes Google's **MediaPipe** Pose Estimation to detect human body landmarks in real-time video.
- Distance Calculation: **OpenCV** is employed to draw lines and dots representing the body's center of gravity (C.G) and foot C.G. The system monitors the horizontal (X-axis) distance between these two points.
- Fall Detection: If the distance between the body C.G and foot C.G exceeds 90 pixels (approximately 75% of the subject's height), the system detects a fall.
- A fall count increases when the foot C.G enters specific areas, indicating different fall locations.
- Post-Fall Monitoring: The system continues to track whether the person stands up after a fall, updating the fall count as necessary.



Fall A person's center of gravity deviates from the center line between the two feet in a fall immediately prior to the accident.

What is a Center of Gravity?

Center of Gravity (COG): Imagine your body is a single point where all your weight is balanced. This point is usually somewhere around your belly or lower chest. If this point stays above your feet, you're stable and won't fall over.

What Happens During a Fall?

When you're standing or moving, your body's center of gravity (COG) usually stays within the space between your feet. This keeps you balanced. A fall happens when your body's center of gravity moves too far outside this area, meaning it moves beyond the space between your feet. When this happens, your body can't stay balanced, and you start to topple over.

Telegram Bot and Alarm Alert System

At the end, we integrated a real-time alert system using a Telegram bot and an alarm. The Telegram bot, built with the **telebot** library, sends instant alerts with captured images to a specified chat when the models detect instances of fire outbreak, violence and fall (e.g. a person falling due to medical emergency).

Additionally, a Python thread plays an alarm sound using the **winsound** module for immediate on-site notification. This will help the receiver to take timely actions which can be helpful in avoiding casualties.

References

1. https://github.com/onenationonemind1/falling_detection/tree/main
2. https://medium.com/@KaziMushfiq1234/fire-detection-with-python-computer-vision-e55c8fc6fa54
3. https://github.com/mushfiq1998/fire-detection-python-opencv
4. https://www.kaggle.com/code/abduulrahmankhalid/real-time-violence-detection-mobilenet-bi-lstm
5. https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset/data

Thank You!