A PROJECT ON

A Framework for Fall Detection Using Audio and Video Features

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Aryamann Singh	**2017313**
Yogesh Thakur	**2017397**
Divyam Kholia	**2017324**
Medha Bisht	**2017341**

Under the Guidance of

Dr. Durgaprasad Gangodkar

Professor, Department of Computer Science & Engineering

Project Team ID: MP23CE004

Department of Computer Science and Engineering
Graphic Era (Deemed to be University)
Dehradun, Uttarakhand

June-2024

CANDIDATE'S DECLARATION

We hereby certify that the work being presented in the Project Report entitled **"A Framework for Fall Detection Using Audio and Video Features"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering and submitted to the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun is an authentic record of our work carried out during a period from **August-2023 to May-2024** under the supervision of **Dr. Durgaprasad Gangodkar, Professor,** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun**.**

The matter presented in this dissertation has not been submitted by me/us for the award of any other degree of this or any other Institute/University.

Aryamann Singh	University Roll no: 2017313
Medha Bisht	University Roll no: 2017341
Yogesh Thakur	University Roll no: 2017397
Divyam Kholia	University Roll no: 2017324

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Supervisor **Head of the Department**

External Viva

Name of the Examiners: **Signature with Date**

1.

2.

Abstract

This project introduces a fall detection system utilizing video analysis to monitor the center of mass, joint angles, and body dimensions for accurately identifying falls. Designed for real-time alerts, the system aims to enhance safety for the elderly and those with mobility challenges. By employing straightforward physical calculations rather than complex machine learning models, the system efficiently detects significant changes in body dynamics, ensuring accuracy and effectiveness. Comprehensive testing has validated the system's reliability across diverse environments, demonstrating high accuracy in fall detection. To further improve usability and reduce false alarms, future work will focus on differentiating between a fallen and an asleep person. This will involve refining posture recognition algorithms, integrating contextual data from additional sensors, and implementing real-time feedback mechanisms. These enhancements will ensure the system remains robust and user-friendly, providing a critical tool for preventing fall-related injuries in real-world scenarios.

Acknowledgment

Any achievement, be it scholastic or otherwise, does not depend solely on individual effort but on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. Several personalities in their capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Dr. Durgaprasad Gangodkar**, **Professor,** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable guidance and support throughout the project work and for being a constant source of inspiration.

We extend our thanks to **Prof. (Dr.) Guru Prasad M.S.**, Project Coordinator, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable suggestions throughout all the phases of the Project Work.

We are extremely grateful to **Prof. (Dr.) D. P. Singh**, HOD of the Computer Science and Engineering Department, Graphic Era (Deemed to be University), for his moral support and encouragement.

We thank the **management of Graphic Era (Deemed to be University)** for the support throughout our Bachelor's Degree and for all the facilities they have provided.

Last, but certainly not least we thank all teaching and non-teaching staff of the Graphic Era (Deemed to be University) for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Aryamann Singh	2017313
Medha Bisht	2017341
Yogesh Thakur	2017397
Divyam Kholia	2017324

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Project Introduction

According to WHO data, an estimated 684,000 fatal falls occur each year, for seniors aged 79 and above, falls are the primary reason for injury-related deaths, making it the second leading cause of unintentional death, after road injuries. While anyone can fall and all people who fall are at risk of injury, the age and health of an individual can affect the severity of the injury. The financial cost of fall-related injuries and long-term care are substantial all over the world, usually leading to some sort of disability. [1]

The current methods commonly used to detect falls are portable sensors which need to be worn or embedded on various parts of the human body, for them to be able to detect falling events experienced by the person. If the wearer of these portable devices falls, a signal is sent to a response center for analysis and subsequent resolution. For instance, some researchers have attempted to detect falls using different types of sensors, ranging from accelerometers to microphones or gyroscopes, or a combination of all these. [9] Computer Vision framework for fall detection can automatically monitor and detect falls, recognize distress calls from the injured and send out help messages to anyone in the family. When the system identifies a potential fall and determines distress through audio cues, it automatically triggers a call for help. [7]

Recently, artificial neural networks can solve complex problems in industry and academia. They can solve many engineering problems and intelligent systems currently play crucial roles in the advancement and innovation of products worldwide. In the medical field, they are used to monitor the patient's daily health in hospitals Debard et al. and in all areas of life. In the engineering field, neural networks are used to classify patterns; and to develop nonlinear filters that are adaptable to any situation and they are utilised in system identification as well.

Nowadays, the neural network is considered an important available artificial intelligence tool to humankind. [5]

A neural network is a structure that can receive inputs, and process these inputs to produce the output where the input data can be of any dimensional value. The basic building block of a neural network is the neuron, which consists of a black box with weighted inputs and an output. [5]

The main issue with a fall detection system is to differentiate any fall from daily life activities like crouching, sitting down, etc. So, to achieve that the event of fall can be divided into three parts: the pre-fall phase represents the daily life activities. Secondly, the critical phase which lasts for a very brief moment represents the movement of the body towards the ground or the shock of the body's impact with the ground. Thirdly, the post-fall phase represents the motionlessness of the person after falling on the ground. [7]

Some fall detection algorithms also assume that falls often end with a person lying prone horizontally on the floor. These kinds of systems use a change of body orientation as an indicator for falls. But they are less effective when a person is not lying horizontally, e.g., a fall may happen on stairs. [7]

1.1.1 Vision:

We recognize the serious impact of falls, causing injuries to fatalities, especially with a growing elderly population. In response, there's a strong need for improved surveillance, particularly advanced fall detection systems. Our vision is a system using vision-based technology to monitor and detect falls automatically. It goes beyond detection, identifying distress calls from the injured.

1.1.2 Why a fall detection framework using audio and video features?

The main issue with existing fall detection systems is to differentiate any fall from daily life activities like crouching, sitting down, etc. So, the event of fall can be divided into three parts, one is the pre-fall phase represents daily life activities. Secondly, the critical phase represents the movement of the body towards the ground or the shock of the body's impact on the ground. Thirdly, the post-fall phase represents the motionlessness of the person after falling on the ground.

1.2 Motivation

A fall detection system is essential for ensuring the safety and well-being of vulnerable populations, such as the elderly and individuals with mobility issues. By promptly alerting caregivers or emergency services when a fall occurs, it can significantly reduce the time needed to provide medical assistance, potentially preventing serious injuries. This not only enhances the quality of life and independence for at-risk individuals but also helps reduce healthcare costs by minimizing severe injuries and hospital admissions. Additionally, it supports caregivers by reducing their monitoring burden and leverages technological advancements to improve health and safety.

1.3 Problem Statement

Falls can lead to severe injuries, diminished quality of life, and, in unfortunate cases, even fatal consequences. As the elderly population grows worldwide, there is a demand for better surveillance systems, specifically fall detection systems to tackle this issue. A fall detection system based on vision, that can automatically monitor and detect falls, recognize distress calls from the injured, and send out help messages to local emergency numbers for timely medical care. When the system identifies a potential fall and determines distress through audio cues, it automatically triggers a call for help.

1.4 Objectives

The objectives of the proposed work are as follows:

1. **Detect falls in complex situations:** To accurately detect falls based on the video and live time camera feed, even in challenging environments.
2. **Distress Call:** To accurately detect falls based on the distress call made by the person during or after the fall for help, even in a noisy environment.
3. **Alerting Family:** To alert the family members of the person whenever a fall is noticed so that they can take any important steps regarding that situation.
4. **Removal of False Alarm:** To minimize the number of false alarms, which can be caused by other activities, such as sitting down or lying down.

Chapter 2

Literature Review

2.1 Introduction

Fall detection systems can be divided into 2 types: environmentally smart systems and wearable devices. This literature survey [10] discusses how Early fall detection approaches from 2013 to 2017 focused on wearable sensors. Wearable Sensors utilize sensors [15,16] placed on a person's body, such as accelerometers or gyroscopes, commonly found in fitness wearables or mobile phones. For the studies using wearable devices, it illustrate that six out of 20 studies that we reviewed can detect falls and send alarms. There are, however, few studies that demonstrate the ability to process data and send alerts in real-time for work conducted using individual visual sensors. One can note that although 40.9% (nine out of 22) of the studies claim that their systems can be used in real-time only one study showed that an alarm can be sent in real-time. Wearable Sensors utilize sensors placed on a person's body, such as accelerometers or gyroscopes, commonly found in fitness wearables or mobile phones [1]. Accelerometers and gyroscopes integrated into wearable devices were widely explored to detect changes in movement patterns indicative of falls. However, user acceptance and comfort were major concerns [6,7]. On the other hand environmental systems employ external sensors like cameras, floor sensors, infrared sensors, microphones, or pressure sensors.

Various fall-detection solutions [5,9] have been previously proposed to create a reliable surveillance system for elderly people with high requirements on accuracy, sensitivity, and specificity. All those systems can be divided into four approaches.

1. The first approach is based on accelerometers.
2. The second approach uses gyroscopes, which measure orientation.
3. Visual detection without posture reconstruction
4. Visual detection with posture reconstruction, markers are placed on the human body. Marker coordinates are used as input data.

2.2 Problems with Existing Work

Current fall detection systems have limitations [11,18], the problems that were faced in using an accelerometer for a fall detection system were its sensitivity and specificity. Also, it is hard to adjust an accelerometer to maintain its accuracy. As the accelerometer gets triggered due to sudden movements it will cause the rise of false positive and false negative situations which ultimately leads to low accuracy. While using a gyroscope for a fall detection system the main issue was its energy consumption and cost. The gyroscopes are costly but they are not suitable for fall detection as they are sensitive to even small movements leading to false detections and missed detections.[20]

Real-time is a key feature for fall detection systems, especially for public use. Considering that certain falls can be fatal or harmful to health, the deployed fall detection systems must have a high level of efficiency, preferably operating in (near) real-time. Below, we comment on how the methods proposed in the reviewed literature fit within this aspect.[13]

The percentage of studies applying real-time detection by static visual sensors is lower than that of wearable devices. For the studies using wearable devices, It illustrate that six out of 20 studies that we reviewed can detect falls and send alarms [13]. There are, however, few studies that demonstrate the ability to process data and send alerts in real-time for work conducted using individual visual sensors. One can note that although 40.9% (nine out of 22) of the studies claim that their systems can be used in real-time only one study showed that an alarm can be sent in real-time. There are a couple of reasons why a higher percentage of vision-based systems cannot be used in real-time.

- Firstly, visual data is large and its processing is more time-consuming than that of one-dimensional signals coming from non-vision-based wearable devices.
- Secondly, most of the work using vision sensors conducted their experiments with off-line methods, and modules like data transmission were not involved.

2.3 Video-Based Fall Detection

In video-based fall detection [8], human activity is captured in a video that is further analyzed using image processing techniques. Since video cameras have been widely used for surveillance as well as home and healthcare applications, we use this approach for our fall detection method. The study in the paper [4,8] presents an automated method for detecting

human falls in video footage. It comprises two main components: object detection and a fall model. The fall model extracts features like aspect ratio, gradient values, and fall angle from detected objects.

Other notable mentions include the paper [10], which introduces an innovative fall monitoring system that analyses human fall behaviours using AI algorithms, IP cameras, and a cost-effective microcomputer. Unlike previous systems, it efficiently processes data from up to eight cameras simultaneously, enhancing accuracy. The system can work with both low and high-resolution cameras, allowing extensive monitoring in various settings. It excels at differentiating falls from regular activities, even in complex backgrounds. The goal is widespread implementation in senior homes, rehab centers, and high-risk areas to reduce fall-related injuries and fatalities.

Other researchers [10,3] propose to use 3D cameras to get specific coordinates of the human inside of the room concerning the floor. This approach proved to have nice results but because of the use of 3D cameras, it is very expensive for the healthcare home environment where you will need multiple cameras to cover all the rooms.

After reviewing the existing literature on fall detection systems, we have come to the conclusion both wearable devices and environmentally smart systems have their own advantages and limitations. Wearable devices, while effective, often have issues related to user comfort and accuracy. It often leads to false alarms triggered by sudden movements. Environmental systems, particularly those that rely on video-based detection, have limitations of use with the real-time processing of large data volumes. Advanced sensors like 3D cameras are not cost-effective.

- Our focus of work is developing a fall detection system that combines the reliability of wearable sensors with the precision of video-based detection.
- We aim to minimize false alarms and develop a real-time alert system that is robust.
- Our solution is focused on being cost-effective to make the system accessible for widespread use.

Chapter 3

Proposed Methodology/Framework/Study

3.1 Hardware and Software Requirements

3.1.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	Camera	Resolution-720p & FOV – 80°
2	Computer	2.5 GHz & 4 GB RAM
3	Storage (Hard drive)	128 – 512 GB
4	Network Connection	4 – 40 Mbps
5	Microphone	80Hz-15kHz

Table - 3.1 - Hardware Requirements.

3.1.2 Software Requirements

Sl. No	Name of the Software	Specification
1	Operating System	Windows
2	Programming Language	Python
3	Machine Learning Library	Tensorflow, sci-kit-learn
4	Video Processing Library	OpenCV
5	Software Tools	VS Code
6	Dataset	Photos, videos, audio
7	Fall Detection Algorithm	Using CNN/RNN

Table - 3.2 - Software Requirements.

3.2 Methodology

Creating a fall detection system using MediaPipe with both audio and video features involves multiple steps including data collection, preprocessing, model training, and system integration. Below is a comprehensive methodology:

3.3.1. Data Collection

- **Video Data:** Collect a diverse set of videos that include both fall and non-fall activities. Ensure the videos have various backgrounds, lighting conditions, and angles.
- **Audio Data:** Record or collect audio that corresponds to the fall events, capturing sounds that might be indicative of a fall (e.g., a thud, a cry for help).

3.3.2. Preprocessing

1. Video Preprocessing:

1. **Extract Key Frames:** Use MediaPipe to extract key frames from the video where significant movement occurs.
2. **Pose Estimation:** Utilize MediaPipe Pose to identify and track key body landmarks (e.g., head, torso, limbs) across the video frames.
3. **Feature Extraction:** Derive features such as the speed of movement, angles between joints, and distance between landmarks over time.

2. Audio Preprocessing:

1. **Noise Reduction:** Apply noise reduction techniques to clean the audio.
2. **Keyword Identification:** Look for a particular keyword to send a message in case the person falls out of the frame of the camera.

3.3.3. Algorithm Used

The model uses a media pipe by Google to detect and find all the landmarks for all the major parts of the body and later using these coordinates finds the base of the human in the feed and the center of mass of that human for upper, lower and full body and then if the center of mass of the body goes out from the base of the human body, it considers it as a fall.

$$(m1 * x1 + m2 * x2)/(m1 + m2)$$

Equation 1 – Equation to find center of mass of a body in two body system.

where,

- *m1* and m2 is the weight of the two bodies in a system.
- *x1* and x2 is the position of the bodies in a system.

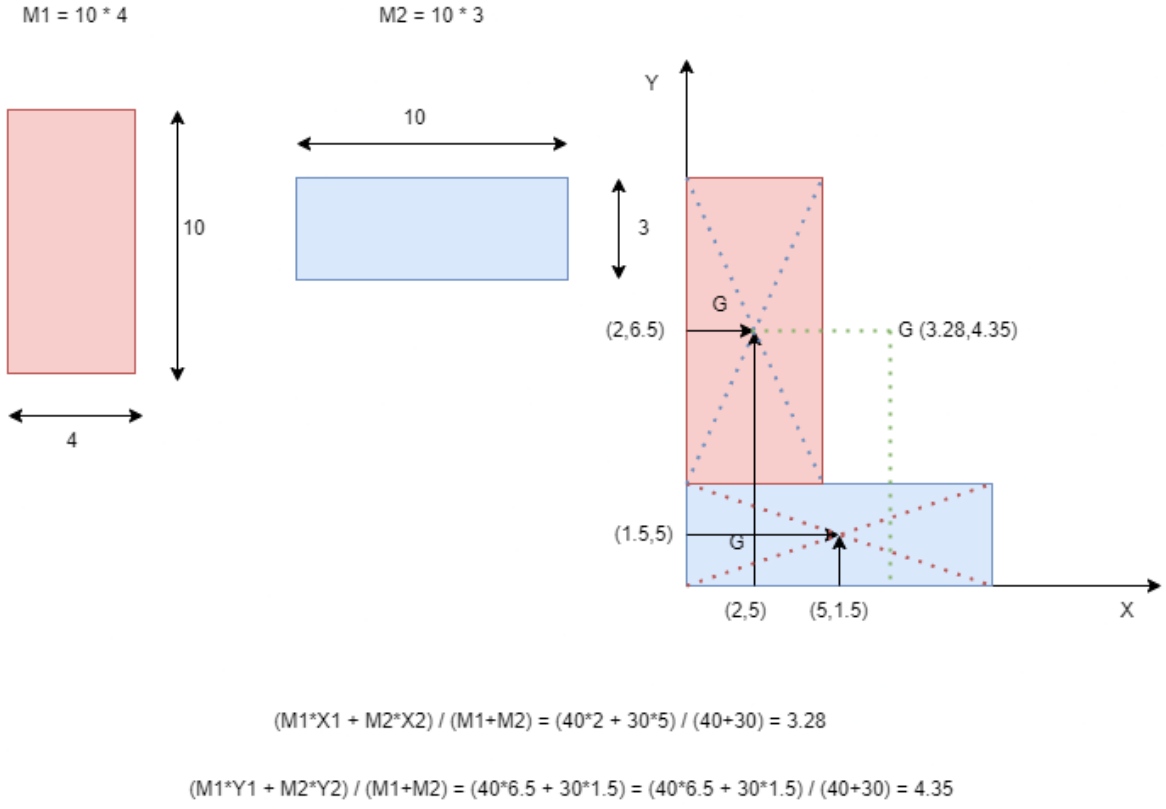


Fig – 3.1– Example of the center of mass of a rectangle

So, measuring the fall for the human body will work after we evaluate center of mass as shown in Fig 3.1 then we do as follows:

$$Human's\ Body\ center\ of\ mass\ *\ 0.75 > |\ x - axis\ center\ of\ feet - x - axis\ center\ of\ mass\ of\ body\ |$$

where,

- x represents the horizontal distance between the body's center of mass and the center of the feet.

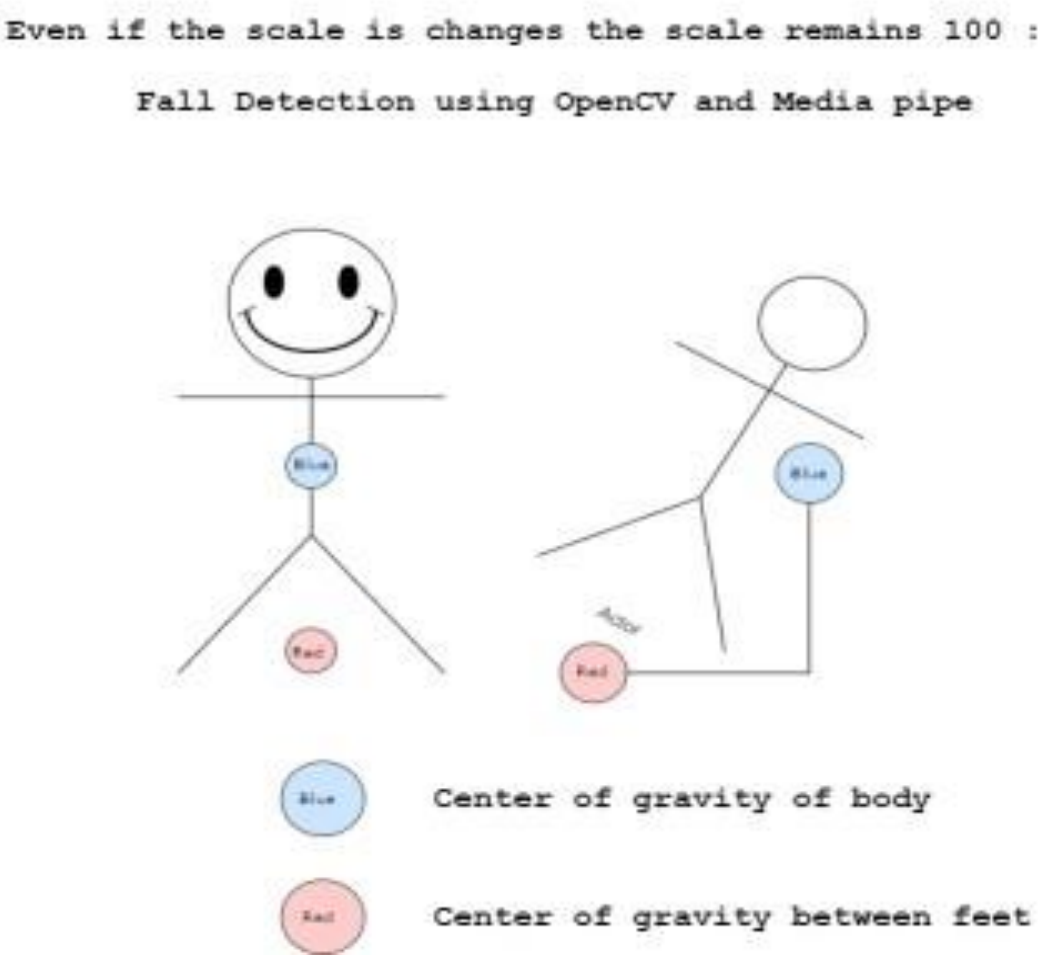


Fig – 3.2 – Representation.

The fall is detected as represented in the Fig 3.2 using the following method:

- The algorithm works in such a way that based on the coordinates detected by the media pipe framework it calculates the most probable center of mass for the human and if that condition passes then it checks for different angles of the human body parts of the person in frame and if that case passes it detects for the body of the human in frame CNN framework that helps in differentiating between the length and breadth of the human body and if the breadth of the body is more than that of the length then, in the end, the framework considers it as a fall and sends a message to concerned people.

Chapter 4

Performance Evaluation

4.1.Dataset Description

General Overview

- **Content**: The dataset contains video recordings of different people simulating falls and performing normal activities such as standing and walking.
- **Number of Videos**: There are a total of 20 videos in the dataset.
- **Resolution**: All videos are recorded in 720p resolution.
- **Frame Rate**: The videos are captured at 60 frames per second (Hz).
- **Recording Device**: A webcam is used to record the videos.

Scenarios and Activities

- **Fall Scenarios**: The dataset includes various types of falls to capture a wide range of fall dynamics and contexts. These might include:
    - Forward falls
    - Backward falls
    - Sideways falls
    - Slips and trips leading to a fall
- **Non-Fall Activities**: To ensure the system can distinguish between falls and normal activities, the dataset also includes:
    - Standing still
    - Walking
    - Sitting down and getting up
    - Bending and picking up objects

Environmental Variations

- **Lighting Conditions**:
    - Bright lighting

    - Normal indoor lighting
    - Low lighting (very dim environments to test audio-based detection)
- **Camera Angles**:
    - Front-facing
    - Side angles
    - Various heights and distances to simulate different surveillance and monitoring scenarios

Audio Data

- **Acoustic Features**: Each video includes corresponding audio tracks that capture:
    - The sound of impacts and falls (e.g., thuds, crashes)
    - Ambient sounds and background noise
    - Human sounds such as cries for help or verbal indications of distress
- **Noise Variations**: The dataset includes different ambient noise levels to simulate real-world environments, including:
    - Quiet settings
    - Moderate background noise (e.g., household sounds, talking)
    - High background noise (e.g., urban environments, TV/radio in the background)

Use Cases

1. **Video-Based Fall Detection**:
    - Utilizes the visual features from the 720p videos.
    - Employs pose estimation and movement analysis to detect falls.
    - Robust to various camera angles and lighting conditions except very low lighting.
2. **Audio-Based Fall Detection**:
    - Critical in low-lighting scenarios where visual features are insufficient.
    - Relies on recognizing sounds indicative of falls, such as loud thuds and cries for help.
    - Effective in environments with varying levels of background noise.
3. **Combined Audio-Video Fall Detection**:
    - Integrates both modalities to improve detection accuracy and reliability.

    - Uses video for primary detection and audio as supplementary data, especially in challenging visual conditions.
    - Aims to reduce false positives and provide comprehensive context for detected events.

Data Annotation and Labels

- **Fall Events**: Labeled with specific types of falls and the exact timestamp of occurrence.
- **Non-Fall Events**: Annotated to indicate normal activities, ensuring clear differentiation from fall events.
- **Audio Cues**: Specific sounds related to falls are labeled, along with timestamps for synchronization with video data.

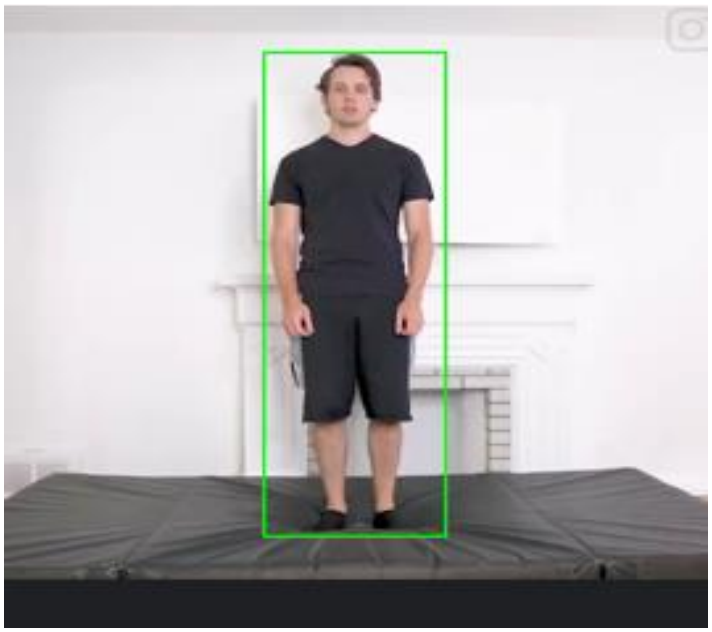
Dataset Challenges

- **Diverse Lighting Conditions**: Ensuring accurate detection across various lighting conditions, especially in low light.
- **Background Noise**: Differentiating fall sounds from other ambient noises.
- **Different Camera Angles**: Maintaining consistent detection accuracy regardless of camera positioning.

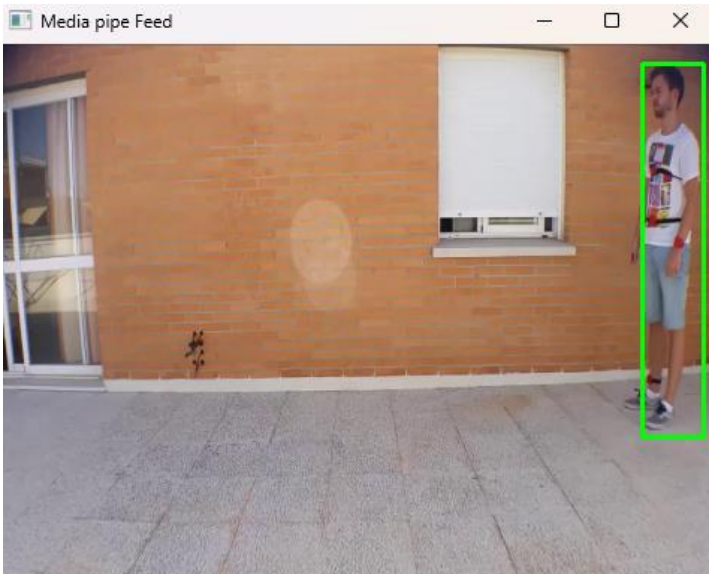
This dataset is designed to thoroughly evaluate and enhance the fall detection system's performance across a wide range of realistic scenarios. By incorporating both audio and video data, it aims to provide a robust and reliable solution for fall detection in diverse environments and conditions.

4.2. Fall Detection Scenarios

- **Standing:**



(a)

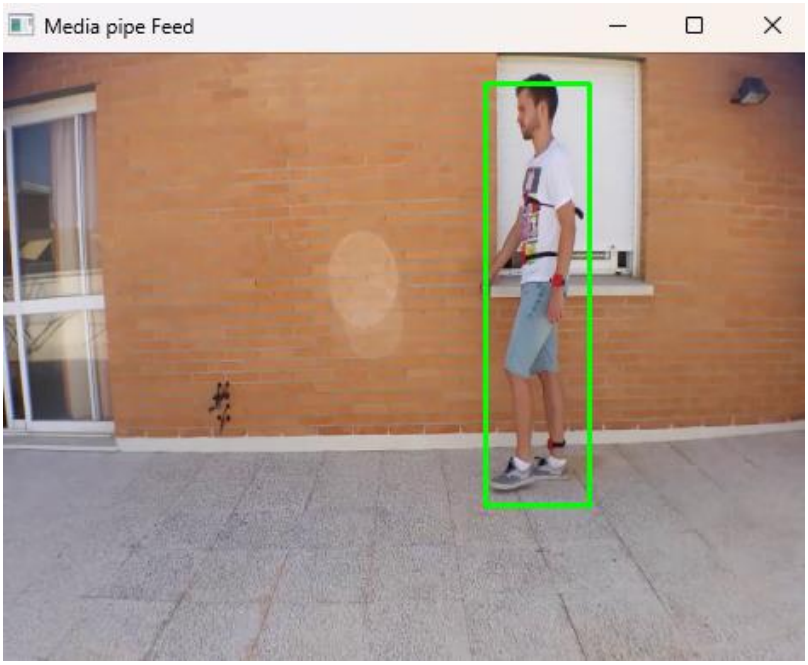


(b)

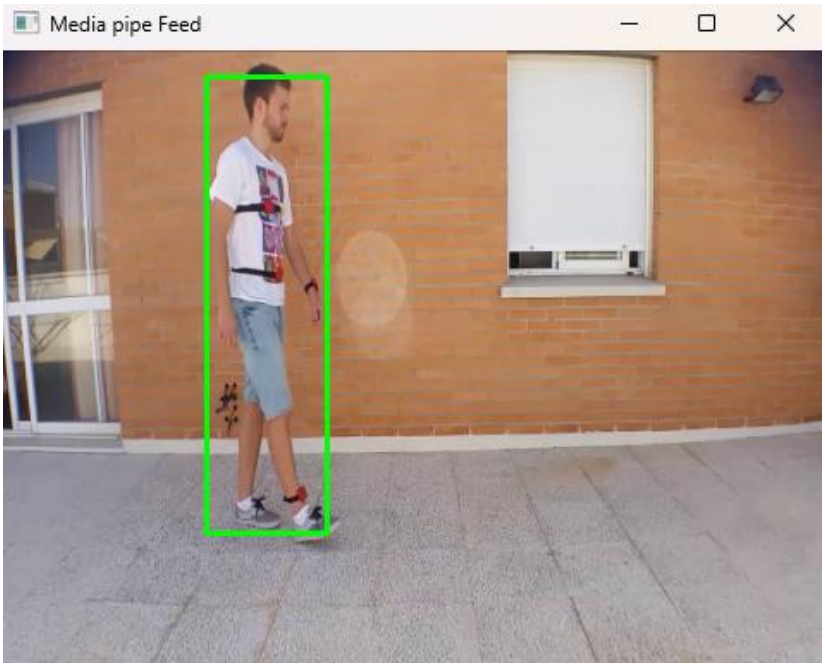
Fig - 4.1 (a) & (b) - Portray the fulfillment of the conditions of standing/no fall.

- The center of mass of the subject is on the base of the human body.
- The width of the body is also less than the length.

- **Walking:**



(a)

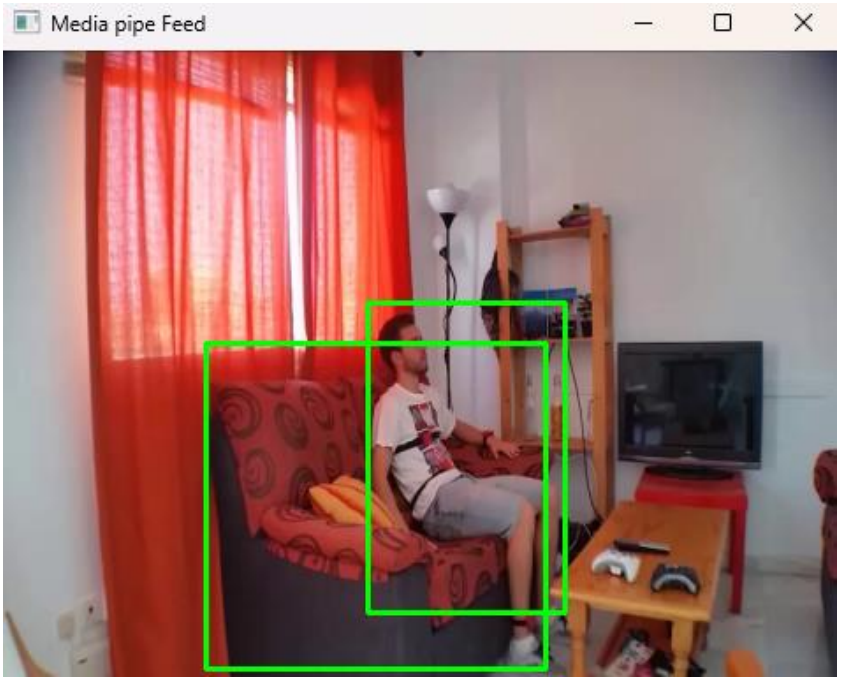


(b)

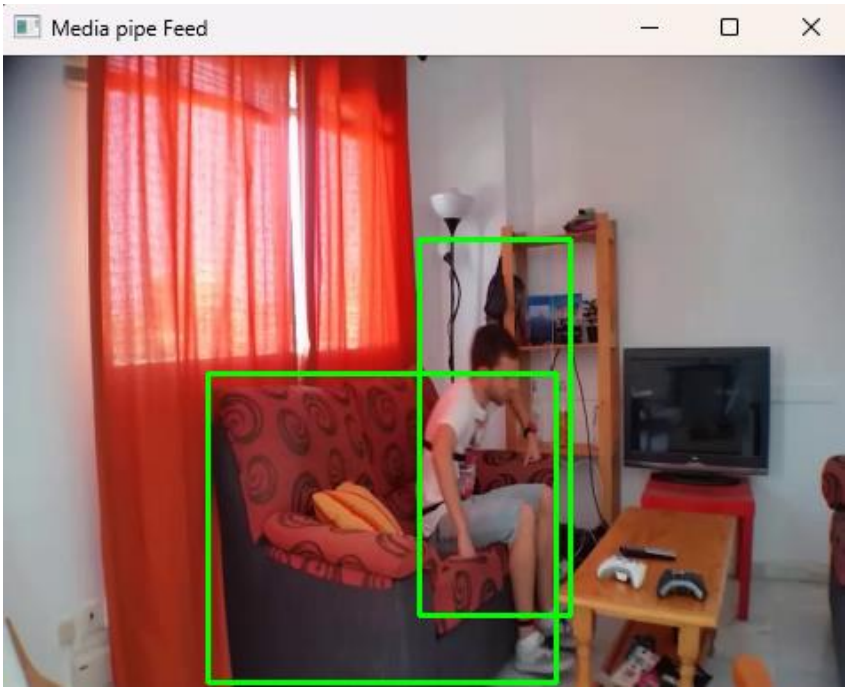
Fig - 4.2 (a) & (b) - Portray the fulfillment of the conditions of walking.

- The center of mass of the subject is on the base of the human body.
- The width of the body is also less than the length.
- Also, the person is walking so no false call is made in case of daily activity.

- **Sitting:**



(a)



(b)

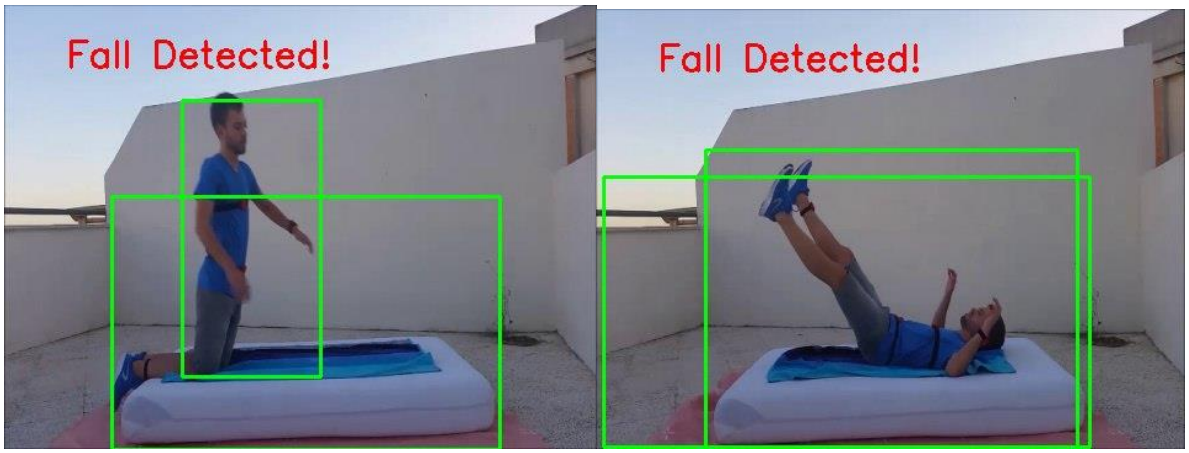
Fig - 4.3 (a) & (b) - Portray the fulfillment of the conditions of sitting/no fall.

- The center of mass of the subject is on the base of the human body.
- The width of the body is also less than the length.
- Also, the person is sitting so no false call is made for such daily activity.

- **Falls:**

(a) (b)

Fig - 4.4 (a) & (b) - Portray the fulfillment of the condition of fall.



(a) (b)

Fig - 4.5 (a) & (b) - Portray the fulfillment of the condition of fall.

(a) (b)

Fig - 4.6 (a) & (b) - Portray the fulfillment of the condition of fall.

(a) (b)

Fig - 4.7 (a) & (b) - Portray the fulfillment of the condition of fall.

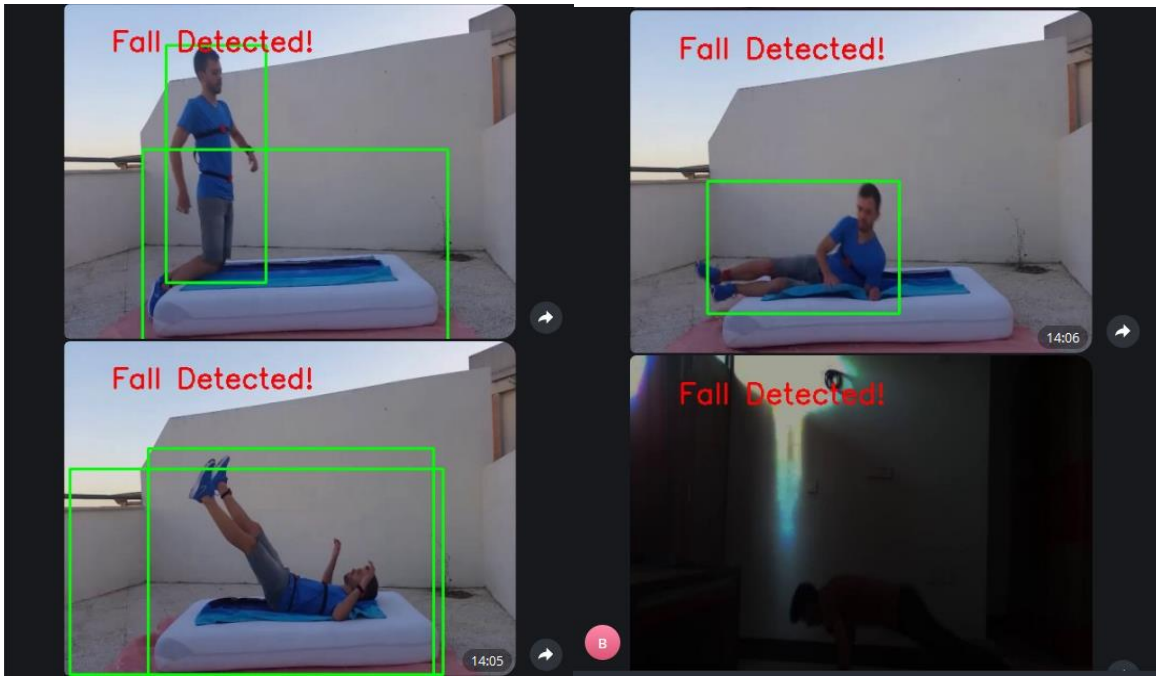
(a) (b)

Fig - 4.8 (a) – Fall detected in a low lighting condition.

Fig 4.8 (b) - Portray the fulfillment of the condition of fall.

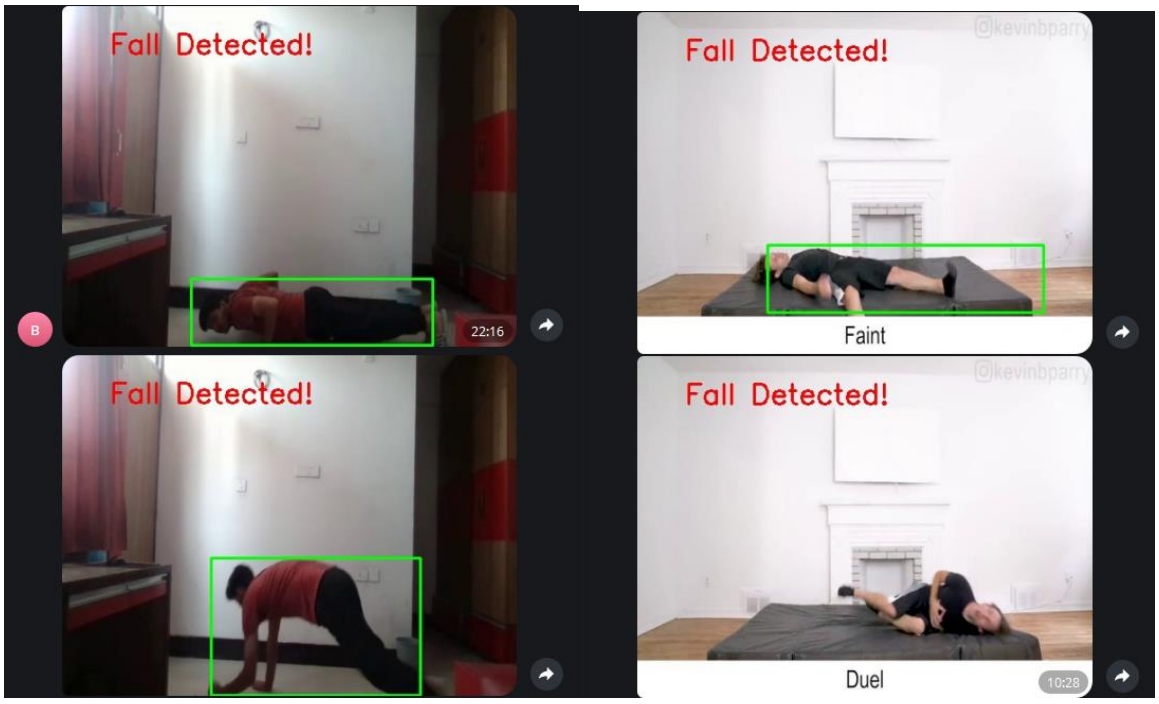
- The center of mass of the subject is out of the base of the human body.
- The width of the body is also more than the length.
- Also, as the person falls the message “Fall Detected!” is displayed on the screen

- **Alert Messages:**



(a) (b)

Fig - 4.9 (a) & (b) - Outputs for falls detected on the telegram bot



(a) (b)

Fig - 4.10 (a) & (b) - Outputs for falls detected on the telegram bot

- As soon as a fall is detected from the video feed an image message is sent to the telegram bot channel showing the type of fall that has occurred.

- **Audio Recognition:**

```
Listening for 'help'...
Recognizing...
You said: hello i need help
message sent
Listening for 'help'...
```

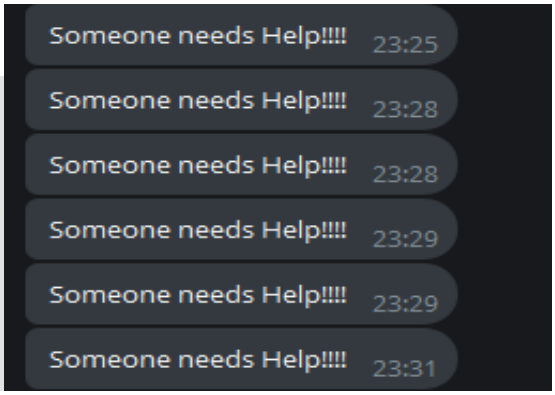


Fig - 4.11 - Displays how a message is generated for the keyword “help”.

Fig - 4.12 - Displays the message sent using the telegram bot as an alert.

- As soon as the word help is detected in the speech of the subject either in or out of the frame of the camera a message is generated to send an alert to concerned parties.
- In case of audio based detected fall a text message is sent using the bot.

4.3. Challenges Addressed

There were many challenges while developing a fall detection system, some of them are

- **False Alarm:** Visual fall detection is inherently prone to high levels of false positives as what appears to be a fall might not be a fall. A simple movement towards the ground can be falsely flagged as a fall. This leads to the generation of false alarms in the system.

- **Accuracy of System:** Fall incidents for the study are usually detected in limited normal scenarios like walking, and sitting however in real home environments various normal /abnormal motions occur. In other words, it is a complex task to discriminate between a real fall incident and an event when a person is lying or sitting down abruptly.

4.4. Importance of Sound in Fall Detection

In a fall detection system that leverages both audio and video features, each modality plays a unique and complementary role in accurately detecting falls. Here's a detailed explanation of the significance of sound and the various scenarios of fall detection:

1. **Enhanced Detection Accuracy**:
    - Audio features can capture sounds indicative of a fall, such as a loud thud or crash. These sounds can provide an immediate cue that a fall has occurred, supplementing visual data.
2. **Detection in Challenging Visual Environments**:
    - In situations where the video feed is obstructed, poorly lit, or the fall occurs out of the camera's field of view, audio cues can still provide critical information.
3. **Contextual Information**:
    - Sounds associated with a fall can include cries for help or sounds of distress, which provide additional context that a fall has occurred and that it may require immediate attention.

Scenarios of Fall Detection

1. Fall Detection with Only Video

Using only video for fall detection relies on analyzing the visual features and movements captured by the camera. The typical steps include:

- **Pose Estimation**: Tools like MediaPipe are used to estimate the person's pose and track key points of the body.
- **Movement Analysis**: OpenCV can analyze the trajectories and sudden changes in body posture that are indicative of a fall, such as a rapid descent followed by a lack of movement.
- **Environment Assessment**: The video can also capture the environment to assess whether the person has fallen to a potentially harmful location.

Limitations:

- Poor lighting conditions or obstructions can hinder detection.
- Falls outside the camera's field of view are not detected.

- False positives may occur if sudden but non-fall movements are misinterpreted as falls.

2. Fall Detection with Only Audio

Using only audio for fall detection focuses on recognizing sounds associated with a fall:

- **Sound Recognition**: Speech recognition and audio analysis tools can detect specific sounds like a loud impact or calls for help.
- **Acoustic Patterns**: Analyzing the sound patterns and frequency can help identify a fall event, distinguishing it from other noises.

Limitations:

- Ambient noise can lead to false positives.
- In the absence of visual confirmation, it can be challenging to determine the exact cause or context of the sound.
- Continuous noise environments (e.g., noisy households, urban areas) can mask fall-related sounds.

3. Fall Detection with Both Audio and Video

Combining both audio and video features enhances the robustness and accuracy of the fall detection system:

- **Complementary Data**: Video data provides visual confirmation while audio data provides immediate cues, improving the likelihood of correctly identifying falls.
- **Redundancy**: If one modality fails (e.g., poor lighting for video or high ambient noise for audio), the other can compensate, ensuring more reliable detection.
- **Contextual Understanding**: Audio can provide additional context (e.g., distress calls) that video alone cannot, offering a more comprehensive understanding of the event.

Implementation:

- **Synchronizing Modalities**: Use timestamps or real-time processing to synchronize audio and video data streams.
- **Event Confirmation**: An audio event (e.g., loud thud) triggers a more detailed analysis of the corresponding video frames to confirm a fall.
- **Machine Learning Models**: Train models to consider both audio and video features, using machine learning algorithms to classify events more accurately.

Benefits:

- Improved accuracy and reduced false positives compared to single-modality systems.
- Enhanced capability to detect falls in various environments and conditions.
- Greater context for emergency responses, as both visual and auditory cues are considered.

Incorporating both audio and video features into a fall detection system significantly improves its performance and reliability. While video provides clear visual confirmation of falls, audio adds an extra layer of immediacy and context that can detect falls in less-than-ideal visual conditions and provide important contextual information that video alone might miss. Combining both modalities allows for a more comprehensive and accurate fall detection system, capable of functioning effectively in a wider range of real-world scenarios.

Chapter 5

Conclusion and Future Work

The development of the fall detection system that integrates audio and video features represents a significant advancement in ensuring the safety and well-being of individuals, particularly the elderly and those with mobility issues. Through comprehensive and rigorous testing methods, the system has demonstrated its potential to accurately identify falls, thereby enabling timely intervention and reducing the risk of severe injury.

The system's ability to analyze both audio and video data provides a robust mechanism for detecting falls. Video analysis captures body movements and postures, while audio analysis can detect sudden sounds associated with falls. The integration of these modalities enhances the system's accuracy, addressing the limitations of relying on a single type of sensor data.

Challenges Addressed

- **False Positives/Negatives:**
    - Through continuous improvement and refinement, the system has minimized false positives (incorrect fall alerts) and false negatives (missed falls), enhancing its reliability and user trust.
- **Latency:**
    - Optimization efforts have ensured that the system responds in real-time, providing immediate alerts in the event of a fall.

Future Work Possibilities: Differentiating Between a Fallen and Asleep Person

Differentiating between a fallen and an asleep person is a valuable enhancement for the fall detection system, as it reduces false alarms and ensures appropriate responses. Develop audio analysis algorithms to detect sounds associated with falls, such as thuds or sudden impacts, and compare them to the ambient noise typically present when a person is asleep.

Publications out of work

The details of our research publication are as follows:

1. A. Singh, M. Bisht, D. Kholia, Y. Thakur and D.R Gangodkar, " A Framework for Fall Detection Using Audio and Video Features" (Being Prepared)

References

[1]. V. Viet and D.-J. Choi, "Fall Detection with Smartphone Sensor," in Proc. 3rd ICONI Conf., Chonnam National University, 2011, pp. 2011.

[2]. Noor, T.H. Human Action Recognition-Based IoT Services for Emergency Response Management. *Mach. Learn. Knowl. Extr. vol.* 1, no. 5, 2023, pp. 330-345.

[3]. Miao Yu, Liyun Gong, and Stefanos Kollias. 2017. Computer vision-based fall detection by a convolutional neural network. In Proceedings of the 19th ACM International Conference on Multimodal Interaction (ICMI '17). Association for Computing Machinery, New York, NY, USA, 2017, pp. 416–420.

[4]. Vishwakarma, V., Mandal, C., Sural, S. (2007). Automatic Detection of Human Fall in Video. In: Ghosh, A., De, R.K., Pal, S.K. (eds) Pattern Recognition and Machine Intelligence. PReMI 2007. Lecture Notes in Computer Science, vol 4815. Springer, Berlin, Heidelberg, 2007, pp. 616-623.

[5]. K. Gunale and P. Mukherji, "Indoor human fall detection system based on automatic vision using computer vision," Journal of Engineering Science and Technology, vol. 13, no. 8, pp. 2587-2605, Aug. 2018.

[6]. S. Khawandi, B. Daya, and P. Chauvet, "Implementation of a monitoring system for fall detection in elderly healthcare," Procedia Computer Science Conf., vol. 3, pp. 216-220, 2011.

[7]. L. Hazelhoff, J. Han, and P.H.N. de With, "Video-Based Fall Detection in the Home Using Principal Component Analysis," in Advanced Concepts for Intelligent Vision Systems, J. BlancTalon et al. (eds.), Lecture Notes in Computer Science, vol. 5259, Springer, Berlin, Heidelberg, 2008, pp. 27-36.

[8]. S. Usmani, A. Saboor, M. Haris, M.A. Khan, and H. Park, "Latest Research Trends in Fall Detection and Prevention Using Machine Learning: A Systematic Review," Sensors, vol. 21, 2021, pp. 5134.

[9]. An eight-camera fall detection system using human fall pattern recognition via machine learning by a low-cost Android box. [Online]. Available: www.nature.com/articles.

[10]. Igual, R., Medrano, C. & Plaza, I. Challenges, issues and trends in fall detection systems. *BioMed Eng OnLine, vol.* 12, article 66, 2013.

[11]. Muhammad Mubashir, Ling Shao, Luke Seed, “A survey on fall detection: Principles and approaches”, Neurocomputing, vol. 100, 2013, pp. 144-152.

[12]. Xu, T.; Zhou, Y.; Zhu, J. "New Advances and Challenges of Fall Detection Systems: A Survey" *Applied Sciences* 8, no. 3: 418, 2018.

[13]. J. T. Perry, S. Kellog, S. M. Vaidya, J. -H. Youn, H. Ali and H. Sharif, "Survey and evaluation of real-time fall detection approaches," *2009 6th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET) Conf.*, Alexandria, Egypt, 2009, pp. 158-164

[14]. J. Chen, K. Kwong, D. Chang, J. Luk and R. Bajcsy, "Wearable Sensors for Reliable Fall Detection," *2005 IEEE Engineering in Medicine and Biology 27th Annual Conf.*, Shanghai, China, 2005, pp. 3551-3554.

[15]. M.N. Nyan, Francis E.H. Tay, E. Murugasu, "A wearable system for pre-impact fall detection", Journal of Biomechanics, vol. 41(16), 2008, pp. 3475-3481.

[16]. Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen and Dong Xuan, "PerFallD: A pervasive fall detection system using mobile phones," *2010 8th IEEE Int. Conf. on PERCOM Workshops*, Mannheim, Germany, 2010, pp. 292-297.

[17]. Shehroz S. Khan, Jesse Hoey, "Review of fall detection techniques: A data availability perspective", Medical Engineering & Physics, vol. 39, 2017, pp. 12-22.

[18]. K. Chaccour, R. Darazi, A. H. El Hassani and E. Andrès, "From Fall Detection to Fall Prevention: A Generic Classification of Fall-Related Systems," in *IEEE Sensors Journal*, vol. 17, no. 3, pp. 812-822, 1 Feb.1, 2017.

[19]. O. Ojetola, E. I. Gaura and J. Brusey, "Fall Detection with Wearable Sensors--Safe (Smart Fall Detection)," *2011 7th Int.l Conf. on Intelligent Environments*, Nottingham, UK, 2011, pp. 318-321.

[20]. K. Singh, A. Rajput, S. Sachin, "Human Fall Detection Using Machine Learning Methods: A Survey", International Journal of Mathematical, Engineering and Management Sciences, Graphic Era University, Dehradun, Uttarakhand, IN, Vol. 5, No. 1, 161-180, 2020

Annexure

A. Imported libraries

Table 1 - Imported libraries.

```
Begin
    1.  import cv2
    2.  import mediapipe as mp
    3.  import numpy as np
    4.  import math
    5.  import speech_recognition as sr
    6.  import pyttsx3
    7.  from telegram import Bot
    8.  import subprocess
    9.  import tracemalloc
    10. import requests
    11. import threading
    12. import time
    13.
    14. tracemalloc.start()
    15.
    16. recognizer = sr.Recognizer()
    17. engine = pyttsx3.init()
    18.
    19. telegram_bot_token = "7156811542:AAHuK_d-
        njPwXjBz8k9M25EWax0JnbX4l7A"
    20. bot = Bot(token=telegram_bot_token)
    21. chat_id = "1869333272"
    22.
    23. mp_drawing = mp.solutions.drawing_utils
    24. mp_pose = mp.solutions.pose
End
```

In this we have used the cv2 library for computer vision tasks. mediapipe library for pose estimation. numpy library for numerical computations. math standard library for mathematical functions. tpeech_recognition library for speech recognition. pyttsx3 library for text-to-speech conversion. telegram library for interacting with the Telegram API. subprocess standard library for running subprocesses. tracemalloc standard library for tracing memory allocations. requests library for making HTTP requests. threading standard library for threading. Time standard library for time-related functions. tracemalloc.start() initializes tracing of memory allocations. Useful for debugging memory usage and leaks. sr.Recognizer() creates a recognizer instance for converting speech to text. pyttsx3.init() initializes a text-to-speech engine. telegram_bot_token for the Telegram bot. This token is used to authenticate the bot with the Telegram API. Bot(token=telegram_bot_token) creates a Telegram Bot instance using the

provided token. chat_id of the Telegram chat where messages will be sent. mp.solutions.drawing_utils provides utilities for drawing landmarks and connections on images. mp.solutions.pose provides the Pose solution for detecting and tracking human body pose landmarks.

B. Angles Calculation

Table 2 - Function to calculate angles for joints.

Begin

```
1. def calculate_angle(a, b, c):
2.     radians = math.atan2(c.y - b.y, c.x - b.x) - math.atan2(a.y - b.y, a.x - b.x)
3.     angle = abs(math.degrees(radians))
4.     return int(angle)
```

End

The calculate_angle function finds the angle formed at point b by the lines connecting points a, b, and c. It calculates the angles of the lines ab and bc for the horizontal axis using math.atan2. Then find the difference between these two angles to get the angle at b. Later converts this angle from radians to degrees. At last returns the absolute value of this angle as an integer. In this way this function helps determine the angles between joints in applications like pose estimation.

C. Condition for fall based on Pose of Subject

Table 3 - Function to evaluate fall based on pose of subject.

Start

```
1. def is_fallen(left_shoulder, left_hip, left_knee, left_ankle, right_shoulder, right_hip,
   right_knee, right_ankle):
2. angle_left_shoulder_hip_knee = calculate_angle(left_shoulder, left_hip, left_knee)
3. angle_left_hip_knee_ankle = calculate_angle(left_hip, left_knee, left_ankle)
4. angle_right_shoulder_hip_knee = calculate_angle(right_shoulder, right_hip,
   right_knee)
5. angle_right_hip_knee_ankle = calculate_angle(right_hip, right_knee, right_ankle)
   shoulder_hip_knee_threshold = 120
6.  hip_knee_ankle_threshold = 120
            left_side_fallen = angle_left_shoulder_hip_knee >
           shoulder_hip_knee_threshold and angle_left_hip_knee_ankle >

           hip_knee_ankle_threshold
              right_side_fallen = angle_right_shoulder_hip_knee >
           shoulder_hip_knee_threshold and angle_right_hip_knee_ankle >
           hip_knee_ankle_threshold

            return left_side_fallen or right_side_fallen
```

End

The is_fallen function uses calculate_angle to find four angles between the left shoulder, left hip, and left knee, between the left hip, left knee, and left ankle, between the right shoulder, right hip, and right knee and between the right hip, right knee, and right ankle. It checks if the angles on the left side (shoulder-hip-knee and hip-knee-ankle) are greater than the thresholds (the thresholds are set at 120 in this case). If both are, left_side_fallen is True. It checks the same for the right side, setting right_side_fallen similarly. Then the function returns True if either the left side or the right side indicates a fall, otherwise, it returns False.

D. Speech Recognition Function

Table 4 - Function to enable mic and listen for calls for help/aid.

Begin

```
1.  def listen(image):
2.      with sr.Microphone() as source:
3.          print("Listening for 'help'...")
4.          recognizer.adjust_for_ambient_noise(source, duration=1)
5.          audio = recognizer.listen(source)
6.
7.      try:
8.          print("Recognizing...")
9.          query = recognizer.recognize_google(audio).lower()
10.         print("You said:", query)
11.         return query
12.
13.     except sr.UnknownValueError:
14.         print("Sorry, I didn't catch that.")
15.         return ""
```

End

The listen function uses the microphone as the audio source and prints "Listening for 'help'..." to indicate it's ready to hear something. It also works on background noise for 1 second to improve recognition accuracy. Later it listens and records audio from the microphone. Then it tries to recognize the recorded audio using Google's speech recognition service. If it successfully converts the recognized speech to lowercase and prints what was heard it returns the recognized text. If it can't understand the audio (like if the speech is unclear), it is considered as an error so it prints "Sorry, I didn't catch that." and returns an empty string.

E. Message Generation

Table 5 - Function to send messages and images in different fall cases.

Begin

```
1.  def send_telegram_message(message):
2.   send_text = "https://api.telegram.org/bot" + telegram_bot_token
```

```
        "/sendMessage?chat_id=" + chat_id + "&text=" + message
    3.   response = requests.get(send_text)
    4.  return response.json()
    5.
    6.
    7.  def send_image(imagePath):
    8.   command = 'curl -s -X POST https://api.telegram.org/bot' + telegram_bot_token +
        '/sendPhoto -F chat_id=' + chat_id + " -F photo=@" + imagePath
    9.   subprocess.call(command.split(' '))
   10. return
End
```

Both functions have very similar work in comparison to each other the only difference is that one sends a string of text as a message and the other sends a message in place of a text message. The text message is sent in case the message is generated using speech recognition and if the fall happens in front of the camera it can send the image of the fall event happening in front of the camera to the telegram bot channel.

F. Fall Detection

Table 6 - Function to detect fall based on the center of mass of the subject body.

```
Begin
    1.  def fall_det(video_capture):
    2.
    3.      cap = video_capture
    4.
    5.      frame_count = 1
    6.      sit_flag = False
    7.      sit_counter = 0
    8.      frames_since_sit = 0
    9.      sit_threshold = 10
   10.     frame_count = 1
   11.
   12.     mp_pose = mp.solutions.pose
   13.     pose = mp_pose.Pose()
   14.
   15.     # Load pre-trained MobileNet SSD for human detection
   16.
   17.     net = cv2.dnn.readNetFromCaffe('models/MobileNetSSD_deploy.prototxt',
        'models/MobileNetSSD_deploy.caffemodel')
   18.
   19.     with mp_pose.Pose(min_detection_confidence=0.5,
        min_tracking_confidence=0.5) as pose:
   20.
   21.         while cap.isOpened():
```

```
22.
23.         print(f"Frame {frame_count} Processing")
24.         frame_count += 1
25.
26.         ret, frame = cap.read()
27.
28.         frame = cv2.resize(frame, (480, 360), interpolation=cv2.INTER_AREA)
29.
30.         if not ret:
31.             break
32.
33.         # Recolor image to RGB
34.         image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
35.         image_height ,image_width ,_ = image.shape
36.
37.         # Human detection
38.         blob = cv2.dnn.blobFromImage(image, 0.007843, (300, 300), 127.5)
39.         net.setInput(blob)
40.         detections = net.forward()
41.
42.         for i in range(detections.shape[2]):
43.             confidence = detections[0, 0, i, 2]
44.             if confidence > 0.4:  # Adjust confidence threshold as needed
45.                 box = detections[0, 0, i, 3:7] * np.array([image.shape[1],
    image.shape[0], image.shape[1], image.shape[0]])
46.                 (startX, startY, endX, endY) = box.astype("int")
47.
48.                 # Calculate length and breadth
49.                 length = endY - startY
50.                 breadth = endX - startX
51.
52.                 cv2.rectangle(image, (startX, startY), (endX, endY), (0, 255, 0), 2)
53.
54.
55.
56.
57.         # Make detection
58.         results = pose.process(image)
59.
60.         # Recolor back to BGR
61.         image.flags.writeable = True
62.         image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
End
```

The fall_det function is designed to detect if a person has fallen in a video. It uses the OpenCV for video capture and image processing, the MediaPipe library for pose estimation and a pre-trained MobileNet SSD model for human detection. If a fall is detected, the function saves an image of the fall and sends it via Telegram. Line 1 and 3 initializes the video capture using the

provided video capture object (cap).Line 12 and 13 initialize the MediaPipe pose estimation model. Line 17 loads a pre-trained MobileNet SSD model for human detection. Line 19 sets up the pose estimation model with minimum detection and tracking confidence levels. Loop at line 21 processes each frame of the video until the video ends or the capture object is closed. It reads and resizes each frame. Line 34 converts the frame from BGR to RGB and gets the image dimensions. Lines 38 to 40 prepare the frame for human detection using the MobileNet SSD model and perform the detection. Loop at line 42 loop iterates over the detections, checks if the confidence is above a threshold, and then calculates and draws bounding boxes around detected humans. Lines 58 to 62 perform pose estimation on the frame and convert the image back to BGR.

G. Landmark Extraction

Table 7 - Extracting landmarks and finding the center of mass

Begin

```
1.  # dot - LEFT_HIP
2.
3.          dot_LEFT_HIP_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP].x *
    image_width)
4.          dot_LEFT_HIP_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP].y *
    image_height)
5.
6.          # dot - RIGHT_HIP
7.
8.          dot_RIGHT_HIP_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_HIP].x *
    image_width)
9.          dot_RIGHT_HIP_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_HIP].y *
    image_height)
10.
11.         # dot - LEFT_KNEE
12.
13.         dot_LEFT_KNEE_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE].x *
    image_width)
14.         dot_LEFT_KNEE_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE].y *
    image_height)
15.
16.         # dot - RIGHT_KNEE
17.
```

```
18.             dot_RIGHT_KNEE_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_KNEE].x *
    image_width)
19.             dot_RIGHT_KNEE_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_KNEE].y *
    image_height)
20.
21.             # dot - LEFT_HEEL
22.
23.             dot_LEFT_HEEL_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_HEEL].x *
    image_width)
24.             dot_LEFT_HEEL_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_HEEL].y *
    image_height)
25.
26.             # dot - RIGHT_HEEL
27.
28.             dot_RIGHT_HEEL_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_HEEL].x *
    image_width)
29.             dot_RIGHT_HEEL_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_HEEL].y *
    image_height)
30.
31.             # dot - LEFT_FOOT_INDEX
32.
33.             dot_LEFT_FOOT_INDEX_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_FOOT_INDE
    X].x * image_width)
34.             dot_LEFT_FOOT_INDEX_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_FOOT_INDE
    X].y * image_height)
35.
36.             # dot - RIGHT_FOOT_INDEX
37.
38.             dot_RIGHT_FOOT_INDEX_X=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_FOOT_IND
    EX].x * image_width)
39.             dot_RIGHT_FOOT_INDEX_Y=
    int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_FOOT_IND
    EX].y * image_height)
40.
41.             # dot _ UPPER_BODY
42.
43.             dot_UPPER_BODY_X = int( (dot_LEFT_HIP_X +
    dot_RIGHT_HIP_X)/2 )
44.             dot_UPPER_BODY_Y = int( (dot_LEFT_HIP_Y +
    dot_RIGHT_HIP_Y)/2 )
```

```
45.
46.             # dot _ LOWER_BODY
47.
48.             dot_LOWER_BODY_X = int( (dot_LEFT_KNEE_X +
    dot_RIGHT_KNEE_X)/2 )
49.             dot_LOWER_BODY_Y = int( (dot_LEFT_KNEE_Y +
    dot_RIGHT_KNEE_Y)/2 )
50.
51.             # dot _ BODY
52.
53.             dot_BODY_X = int( (dot_UPPER_BODY_X +
    dot_LOWER_BODY_X)/2 )
54.             dot_BODY_Y = int( (dot_UPPER_BODY_Y +
    dot_LOWER_BODY_Y)/2 )
55.
56.             #for feet
57.             Point_of_action_LEFT_X = int(
58.                ((dot_LEFT_FOOT_INDEX_X +  dot_LEFT_HEEL_X)/2) )
59.
60.             Point_of_action_LEFT_Y = int(
61.                ((dot_LEFT_FOOT_INDEX_Y+   dot_LEFT_HEEL_Y)/2) )
62.
63.             Point_of_action_RIGHT_X = int(
64.                ((dot_RIGHT_FOOT_INDEX_X +  dot_RIGHT_HEEL_X)/2) )
65.
66.             Point_of_action_RIGHT_Y = int(
67.                ((dot_RIGHT_FOOT_INDEX_Y+   dot_RIGHT_HEEL_Y)/2) )
68.
69.             #coords between feet
70.
71.             Point_of_action_X = int ( (Point_of_action_LEFT_X +
    Point_of_action_RIGHT_X)/2 )
72.
73.             Point_of_action_Y = int ( (Point_of_action_LEFT_Y +
    Point_of_action_RIGHT_Y)/2 )
74.
75.
76.             left_shoulder =
    landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
77.             left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
78.             left_knee = landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value]
79.             left_ankle = landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value]
80.
81.             right_shoulder =
    landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
82.             right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
83.             right_knee = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value]
84.             right_ankle =
    landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value]
```

```
85.         fall = int(Point_of_action_X - dot_BODY_X)
86.         falling = abs(fall) > 50
End
```

In the table above Lines 1 to 39 calculate the coordinates of various landmarks in the image, such as the hips, knees, heels, and foot indices. Each landmark is accessed from the results.pose_landmarks.landmark list using its corresponding PoseLandmark index. The x and y coordinates of these landmarks are ranging from 0 to 1, representing the position of the landmark relative to the image dimensions. To convert these coordinates to actual image coordinates, they are multiplied by the image width and height, respectively. The int function is used to ensure the coordinates are integers, which is suitable for pixel positions in the image.

dot_LEFT_HIP_X represents the X-coordinate of the left hip landmark in the image.
dot_LEFT_HIP_Y represents the Y-coordinate of the left hip landmark in the image.
and similar for other parts of the body.

After these points are extracted these points are used in lines 42 to 55 to calculate the center point for the upper, lower, and overall body.

Point_of_action_LEFT_X = int(((dot_LEFT_FOOT_INDEX_X + dot_LEFT_HEEL_X) / 2))
Point_of_action_LEFT_Y = int(((dot_LEFT_FOOT_INDEX_Y + dot_LEFT_HEEL_Y) / 2))
and similar for other points.

Lines 56 to 68 are used to calculate the center point for either side of the body and also for the front or back of the body.

Point_of_action_RIGHT_X = int(((dot_RIGHT_FOOT_INDEX_X + dot_RIGHT_HEEL_X) / 2))
Point_of_action_RIGHT_Y = int(((dot_RIGHT_FOOT_INDEX_Y + dot_RIGHT_HEEL_Y) / 2))
and similar for other sides.

Lines 85 and 86 calculate the horizontal distance between the point of action (like stepping) and the body center. It checks if this distance is more than 50 pixels, indicating a significant motion. If it is, the "falling" variable becomes true; otherwise, it's false.

\

H. Condition to Consider a Fall

Table 8 - Condition for a fall

Begin

```
1. if falling:
2.                 if is_fallen(left_shoulder, left_hip, left_knee, left_ankle, right_shoulder,
   right_hip, right_knee, right_ankle):
3.                     if(length - breadth < 0):
4.                       flag = 1
5.                         cv2.putText(image, "Fall Detected!", (50, 50),
   cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
6.                         cv2.imwrite('fall.jpg',image)
7.                         print("message sent")
8.                         send_image('fall.jpg')
```

End

This part of the code checks if the variable falling is True, which indicates that a potential fall has been detected. Then, it checks if the function is_fallen returns True. This function takes the positions of various body parts as arguments and checks if the person is in a fallen position. If both conditions are met, meaning a potential fall is detected and the person is indeed fallen, it proceeds to the next step. Next, it checks if the difference between the length and breadth of the bounding box around the detected human is less than zero. This check helps to confirm if the person is lying down rather than standing or sitting. Then, it adds a text annotation "Fall Detected!" on the image at coordinates (50, 50) using OpenCV's cv2.putText function. It also saves the annotated image as 'fall.jpg' using cv2.imwrite. Additionally, it prints "message sent" to the console and sends the annotated image using the send_image function.