Chapter 1

Introduction and Problem Statement

1.1 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, in order for them to be able to detect falling events experienced by the person. If the wearer of these portable devices falls down, 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 based 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 patients' daily health in hospitals Debard et al. and in all areas of life. In the engineering field, the neural networks are used to classify patterns; to develop nonlinear filters which are adaptable to any situations and they are utilized 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 which can receive inputs, 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 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 on is the pre-fall phase represents the daily life activities. Secondly, the critical phase which last for a very brief moment which represents the movement of body towards the ground or the shock of the body's impact with the ground. Thirdly, the post fall phase representing the motionlessness of the person after falling on the ground. [7]

Some fall detection algorithms also assume that falls of-ten end with a person lying prone horizontally on the floor. These kinds of systems use 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 automatically monitor and detect falls. 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 the daily life activities. Secondly, the critical phase which represents the movement of body towards the ground or the shock of the body's impact with the ground. Thirdly, the post fall phase representing the motionlessness of the person after falling on the ground.

1.2 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.

Chapter 2

Objectives

The objectives of the proposed work are as follows:

1. **Detect fall in complex situation:** To accurately detect falls on the basis of the video and live time camera feed, even in challenging environments.
2. **Distress Call:** To accurately detect falls on the basis of the distress call made by the person during or after the fall for help, even in noisy environment.
3. **Fast Response:** To minimize the delay between the occurrence of a fall and the system's detection for emergency situations.
4. **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.
5. **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 3

Project Work Carried Out

3.1. Research Done so Far:

A fall detection system implemented using Media Pipe and OpenCV works in the following steps:

1. Initialization:

- Capture video stream: OpenCV captures video from a camera.
- Media Pipe setup: Media Pipe's Holistic model is loaded for pose estimation.

2. Frame Processing:

- Frame reading: Each frame is read from the video stream.
- Pose estimation: Media Pipe analyses the frame and generates 3D key points representing the body's pose.
- Feature extraction: Relevant features for fall detection are calculated from the key points, such as height, orientation, velocity, acceleration, and angles between limbs.

3. Fall Detection Logic:

- Feature analysis: The extracted features are compared to predefined thresholds for fall detection.
- Temporal analysis: Features are analysed over multiple frames to differentiate sudden changes from regular movements.
- State machine: (Optional) A state machine tracks the person's overall motion sequence ("standing," "falling," "recovering") for improved accuracy.
- Fall detection: If the features and sequence satisfy the fall detection criteria, a fall event is confirmed.

Here's a simplified breakdown of the workflow:

1. Video frame captured.
2. Pose estimated with Media Pipe.
3. Key point data extracted.

4. Relevant features calculated.
5. Features compared to fall detection thresholds.
6. Temporal analysis performed.
7. Fall confirmed based on features and sequence.
8. Alert triggered, and appropriate action taken.

Technical details:

- Media Pipe: Provides real-time pose estimation with 3D key points.
- OpenCV: Handles video capture, image processing, and thresholding.
- Feature analysis: Specific features chosen and thresholds set depending on the application and desired sensitivity.
- Fall detection logic: Can be simple rule-based or involve more complex machine learning models.

Benefits of using Media Pipe and OpenCV:

- Open source and free to use.
- Easy to set up and implement.
- Flexibility to customize and optimize for specific needs.
- Active community and resources available for support.

3.2 Research Progress Overview:

To estimate human body, pose in a video/image we used media pipe – pose estimation by google. Its land marker task lets us to detect different points of human body in an image or video as well as in live feed. So, to detect a fall the model measures if the center of gravity of a person in the feed is in between the coordinates of the feet else the system detects a fall in the feed.

- **Step 1: Extracting Landmarks**

  Using the media pipe landmark function, we extracted the landmarks for different body parts of a human in the feed. The coordinates for different joints are calculated in different manner the different manners are displayed in the images below.

```
# finding the position of body part

dot_LEFT_SHOULDER_X= int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER].x * image_width)
dot_LEFT_SHOULDER_Y= int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER].y * image_hight)

dot_LEFT_ELBOW_X= int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_ELBOW].x * image_width)
dot_LEFT_ELBOW_Y= int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_ELBOW].y * image_hight)
```

Fig – 3.2.2 – Extracting landmarks.

- **Step 2: Calculating Angles**

Using the landmarks extracted from the last step we calculate the angles between different joints of human body. The method is displayed in images below.

```
# calculating angle

angle_shoulder_l = calculate_angle(elbow_l, shoulder_l, hip_l)
```

Fig – 3.2.3 – Calculating angles using extracted landmarks.

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

Fig – 3.2.4 – Function of calculating angles.

- **Step 3: Extracting Points for Base of Human Body**

This part is most important as we are calculating and generating the coordinates of the base of human body which will be one of the deciding factors of a fall. These points will be extracted using the landmarks extracted in the *Step 1*. These points are denoted as points of action and the way these points of actions are calculated is shown below.

```
# points of actions for both feet

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) )


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) )


# co ordinates between Feet

Point_of_action_X = int ( (Point_of_action_LEFT_X +  Point_of_action_RIGHT_X)/2 )

Point_of_action_Y = int ( (Point_of_action_LEFT_Y +  Point_of_action_RIGHT_Y)/2 )


# co ordinates between feet
Point_of_action = [Point_of_action_X , Point_of_action_Y]
```

Fig -3.2.5 – Points of action calculation.

- **Step 4: Detecting Fall**

  To detect the fall, we subtract the value of x coordinate of base (point of action) with the x coordinate of the center of mass/gravity of the body and then if it meets certain conditions then it is considered as a fall else the person is considered as standing. The conditions are in the image below.

```
fall = int(Point_of_action_X - dot_BODY_X )
falling = abs(fall) > 50
```

Fig – 3.2.6 – Fall Condition on the basis of center of mass.

```
def is_fallen(left_shoulder, left_hip, left_knee, left_ankle, right_shoulder, right_hip, right_knee, right_ankle):
    angle_left_shoulder_hip_knee = calculate_angle(left_shoulder, left_hip, left_knee)
    angle_left_hip_knee_ankle = calculate_angle(left_hip, left_knee, left_ankle)
    angle_right_shoulder_hip_knee = calculate_angle(right_shoulder, right_hip, right_knee)
    angle_right_hip_knee_ankle = calculate_angle(right_hip, right_knee, right_ankle)

    shoulder_hip_knee_threshold = 120
    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
```

Fig – 3.2.7 – Fall Condition on the basis of angles of body parts.

```
if falling:
    if is_fallen(left_shoulder, left_hip, left_knee, left_ankle, right_shoulder, right_hip, right_knee, right_ankle):
        if(length - breadth < 0):
            flag = 1
            cv2.putText(frame, "Fall Detected!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            send_telegram_message("Someone Fell Down!(image)")
            print("message sent")
```

Fig – 3.2.8 – Fall conditions checks before declaring it a fall.

3.3 Algorithm Used:

The model uses 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, then it considers it as a fall.

$$(m1 * x1 + m^2 * x^2)/(m1 + m^2)$$

Fig – 3.3.1 – Formula to calculate rectangle's center of gravity

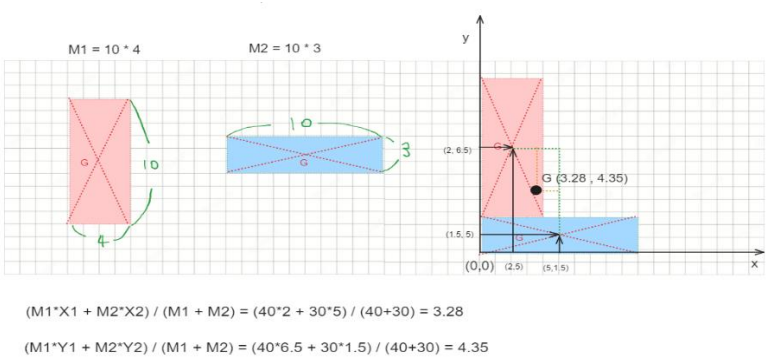

Fig – 3.3.2 – Example of center of mass of a rectangle

So, measuring the fall for human body will work like as follow:
Human's Body center of mass * 0.75 > | x-axis center of feet – x-axis center of mass of body

The fall is detected in the following method:

- The algorithm works in such a way on the basis of 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 dnn 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.

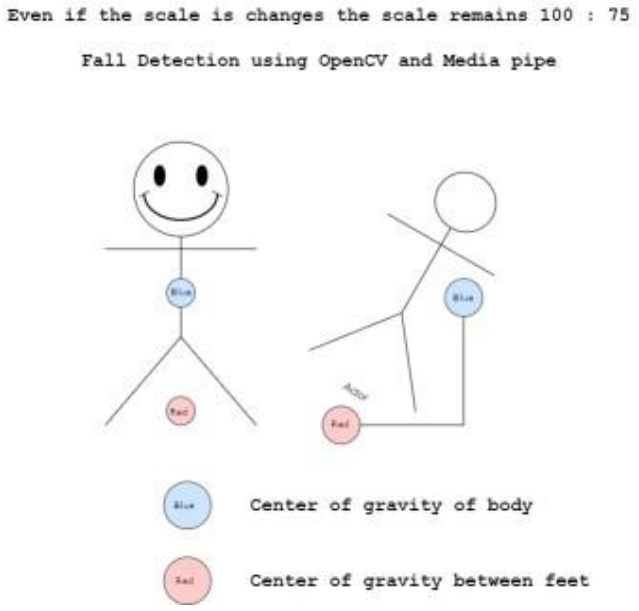

Fig – 3.3.3 – Representation.

3.4 Tools used:

- **Media Pipe** – For pose estimation
- **Open CV** – For using videos/images as input

3.5 Sound Recognition:

To detect distress call made by the person falling in the video frame or even outside the camera frame, the feature of speech recognition has been added so that if a person falls outside the frame and we are not able to see them falling so if they are conscious enough to make a call for help or aid then an alert can be sent to their family to check on them in that case and if the situation folds out to be serious then emergency arrangements can be made for them.

- **Step 1: Listening for call**

  To detect sound the framework, keep the microphone enabled all the time and whatever it hears is added to a string to be processed later the new string is generated every time the is an ambient pause in the audio being picked up by the microphone.

```
def listen():
    with sr.Microphone() as source:
        print("Listening for 'help'...")
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        print("Recognizing...")
        query = recognizer.recognize_google(audio).lower()
        print("You said:", query)
        return query
    except sr.UnknownValueError:
        print("Sorry, I didn't catch that.")
        return ""
```

Fig – 3.5.1 – Function to enable mic and listen for call for help/aid.

- **Step 2: Checking the condition**

  After a string is generated by the framework after listening to the conversation going on around it. It checks for the “help” keywork in the string and if the respective condition is met it sends a message as per the condition.

```
    query = listen()
    if "help" in query:
        send_telegram_message("Someone needs help!(audio)")
```

Fig – 3.5.2 – Condition to check for call for help/aid.

3.6 Message Generation:

After any of the condition is met either it is from the fall detected in the camera feed/video or from the call made for help/aid using the keyword help, the framework send a message using a telegram bot depending on the type of condition met.

```
def send_telegram_message(message):
    send_text = "https://api.telegram.org/bot" + telegram_bot_token
    a+ "/sendMessage?chat_id=" + chat_id + "&text=" + message
    responce = requests.get(send_text)
    return responce.json()
```

Fig – 3.6.1 – Message generation function.

3.7 Results:

- **Fallen:**

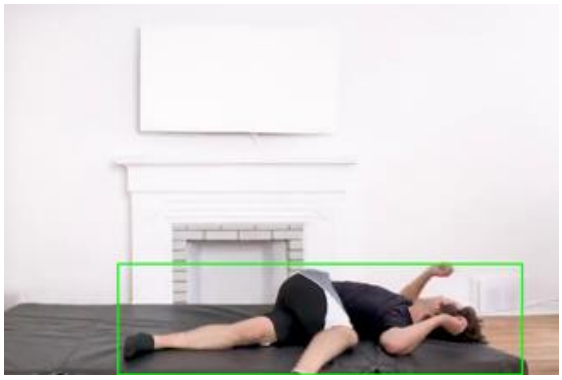

Fig – 3.7.1 & 3.7.2 – Portray the fulfilment of the conditions of fall.

1. As the center of mass of the subject is out of base of human body.
2. The width of the body is more than that of the length.

- **Message sent:**

```
Frame 90 Processing
Frame 91 Processing
Frame 92 Processing
message sent
Frame 93 Processing
message sent
Frame 94 Processing
message sent
```

Fig – 3.7.3 – Displays the Processing of video frames.

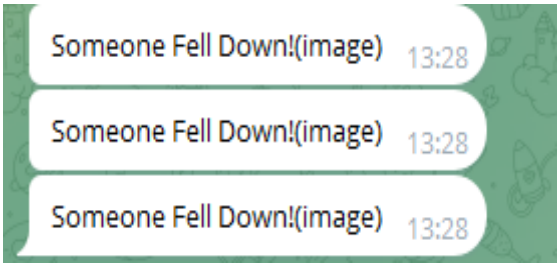

Fig – 3.7.4 – Displays the message sent in case of fall for every frame.

- **Standing:**

Fig – 3.7.5 & 3.7.6 – Portray the fulfilment of the conditions of standing/no fall.

1. As the center of mass of the subject is on the base of human body.
2. The width of the body is also less then the length.

- **Audio Recognition:**

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

Fig – 3.7.7 – Audio Recognition working with microphone enabled in background.

1. Updated the string for recognized words as soon as ambient silence is gained.
2. Sends message when “help” is encountered in the string.

- **Message sent:**

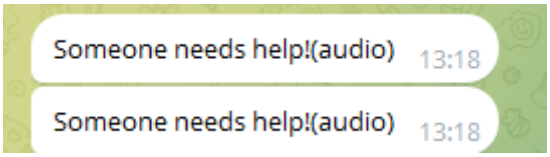

Fig – 3.7.4 – Displays the message sent in case of call for help/aid.

Chapter 4

Future Work Plan

The future work plan of our project are as follows:

Sl. No.	Work Description	Duration in Days
1	We plan to send an image of the fall every time a fall is noticed with the alert message being sent to the concerned parties.	**5 – 7 days**
2	We plan to integrate all the features of this frame work into a single file so that both the functions can work simultaneously.	**7 – 10 days**
3	We plan to reduce the false calls as much as possible for better use and making this model as convenient as possible	**5 – 7 days**

Chapter 5

Weekly Task

The report of project work allocated by the supervisor is as follows:

Week No.	Date: From-To	Work Allocated	Work Completed (Yes/No)	Remarks	Guide Signature
1	7th Jan – 14th Jan	Learning and understanding speech recognition.	Yes		
2	15th Jan - 24th Jan	Implementing speech recognition.	Yes		
3	25th Jan- 31th Jan	Generating alerts as because of condition met in speed recognition.	Yes		
4	1st Feb - 7th Feb	Generating alerts as because of condition met in fall detection.	Yes		
5	7th Feb- 17th Feb	Reducing false calls in fall detection by adding in more conditions	Yes		
6	18th Feb - 26th Feb	Gather feedback and make adjustments to the work done so far	Yes		

Chapter 6

Details of Major Project Research Paper

1.	**Project Team ID**	MP23CE004
2.	**Name of Supervisor**	Dr. Durgaprasad Gangodkar
3.	**Title of the paper**	A Framework for Fall Detection using Audio and Video Features
4.	**Authors Name**	1. Medha Bisht 2. Divyam Kholia 3. Aryamann Singh 4. Yogesh Thakur
5.	**Percentage of plagiarism (Check in Turnitin, 10 words)**	
6.	**Status of the research paper** ***(Project supervisor has to mark the appropriate one)***	• Published/ Presented • Registered for conference • Accepted • Communicated • Not-Communicated
7.	**Scopus Indexed (mark the appropriate one)**	• Journal • Conference
	Signature of students	Signature of Supervisor

References

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

[2] V. Vishwakarma, C. Mandal, and S. Sural, "Automatic Detection of Human Fall in Video," School of Information Technology, Indian Institute of Technology, Kharagpur, India.

[3] R. Shu1 and J. Shu2, "An eight-camera fall detection system using human fall pattern recognition via machine learning by a low-cost android box," www.nature.com/articles.

[4] 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. Blanc-Talon et al. (eds.), Lecture Notes in Computer Science, vol. 5259, Springer, Berlin, Heidelberg, 2008, pp. 27-36.

[5] L. Alhimale, H. Zedan, and A. Al-Bayatti, "The implementation of an intelligent and video-based fall detection system using a neural network," Applied Soft Computing, vol. 18, pp. 59-69, 2014.

[6] 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, 416–420.

[7] 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.

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

[9] T.H. Noor, "Human Action Recognition-Based IoT Services for Emergency Response Management," Mach. Learn. Knowl. Extr., vol. 5, pp. 330-345, 2023.

[10] 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, pp. 5134, 2021.