Methodologies for Fall Detection

**Methodology 01: Fall Detection Using Bounding Box Aspect Ratio**

**1. Introduction**

This approach utilizes object detection to identify a person in a video feed and determines if they have fallen based on the aspect ratio of the bounding box. The method assumes that a standing person has a greater height than width, whereas a fallen individual will have a width comparable to or greater than their height. By analyzing these dimensions in real time, the system can provide alerts for potential falls.

**2. Dataset and Model Selection**

The YOLOv8m model is employed for real-time object detection due to its speed and accuracy. The model is pre-trained on the COCO dataset, which includes person detection in various environments and poses. A class label list is loaded from a text file to ensure detected objects are classified correctly. The pre-trained model is fine-tuned on an industry-specific dataset (if available) to improve accuracy in detecting workers.

**3. Real-Time Video Processing**

A live webcam feed is captured using OpenCV (cv2.VideoCapture). The frame dimensions are set to 980x740 pixels to maintain a consistent field of view and ensure accurate detection. Each frame is processed individually, and the YOLO model is applied to extract bounding boxes and object labels. The frame rate is optimized to ensure low latency detection, allowing near-instantaneous fall detection alerts.

**4. Bounding Box Analysis for Fall Detection**

For each detected person, the bounding box coordinates (x1, y1, x2, y2) are extracted. The height and width of the bounding box are computed as:

Height = y2 – y1

Width = x2 – x1

The aspect ratio of the bounding box is calculated as:

Aspect Ratio = (Height / Width)

A fall detection threshold is defined as the difference between height and width:

Threshold = ext(Height) – ext(Width)

The decision rule for fall detection is:

If Height > Width and Threshold > 0, the person is standing.

If Height ≤ Width and Threshold ≤ 0, the person is likely to have fallen.

A confidence threshold (e.g., 80%) ensures that only high-confidence detections are considered to reduce false positives. Bounding box coordinates and threshold values are constantly updated as new frames are processed to account for movement and posture changes.

**5. Visualization and Alerts**

* Bounding boxes are drawn around detected persons using cvzone.cornerRect() to visually mark detections.
* The confidence level is displayed alongside the detected person to indicate detection reliability.
* If a fall is detected, a red warning label 'Fall Detected' is displayed on the video feed.
* The system continuously processes new frames to reassess fall status in real time.
* Future integrations could include audible alarms or automated notifications to supervisors.

**6. Disadvantages of This Approach**

1. **Inaccuracy in Certain Poses –** A person bending, crouching, or sitting may be misclassified as fallen due to similar aspect ratios.
2. **Camera Angle Dependence –** The model works best with a side-view camera; top-down or oblique angles may distort aspect ratio calculations.
3. **Occlusion Issues –** Objects blocking parts of a person (e.g., machinery, other workers) can lead to incorrect bounding box measurements.
4. **No Temporal Analysis –** The method lacks motion tracking or historical data, which means temporary low postures can cause false positives.
5. **Variation in Body Sizes –** Taller or broader individuals may not fit the predefined aspect ratio rules, reducing reliability.
6. **Environmental Factors –** Poor lighting, shadows, or fast movements may cause detection failures or erratic bounding boxes.

**7. Conclusion and Future Improvements**

This approach provides a simple and efficient method for fall detection in real-time but has limitations due to its reliance on static bounding box dimensions. Future improvements could incorporate:

* Pose estimation models (e.g., OpenPose, MediaPipe) to analyze body posture directly rather than relying solely on bounding box dimensions.
* Optical flow analysis to track motion over multiple frames, reducing false positives.
* Machine learning-based threshold optimization, allowing adaptive fall detection that considers different worker sizes and environments.
* Multi-camera integration, where multiple angles are used to verify falls before triggering alerts.

By enhancing the system with these additional methods, fall detection accuracy can be significantly improved, making it more suitable for real-world industrial applications.

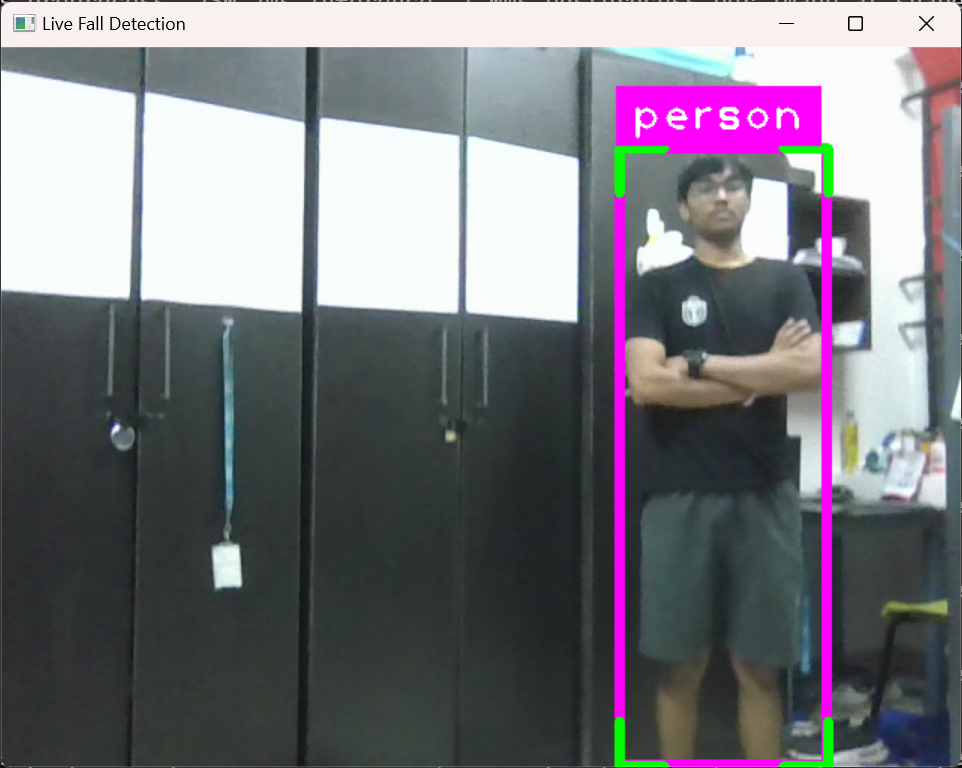
**8. Output of Implementation:**

Figure 01: Standing Person

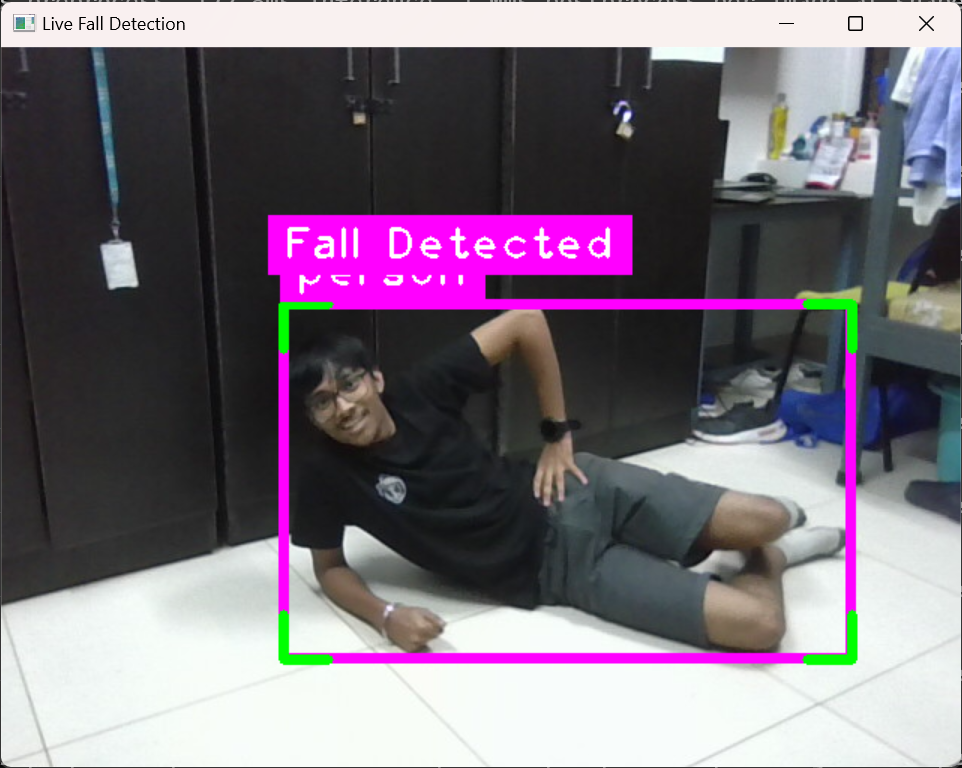


Figure 02: Fall Detected

**Methodology: Fall Detection Using Pose Estimation**

**1. Introduction**

This approach leverages MediaPipe Pose to detect and analyze human posture in real-time. Unlike more advanced methods that utilize vector-based calculations for spine inclination, this approach primarily relies on relative head and shoulder positioning. While simpler, it lacks robustness in differentiating between falls and other similar postures.

**2. Model and Dataset**

* **MediaPipe Pose** is used for landmark detection due to its efficiency.
* The model operates in real-time using OpenCV to capture and process video frames.
* No additional dataset training is required, but results can be improved with domain-specific pose correction models.

**3. Frame Processing and Landmark Extraction**

* The system captures video frames and converts them to **RGB format** for processing.
* MediaPipe extracts **33 key landmarks** corresponding to human joints.
* The key joints used for fall detection:
  + Right Shoulder (11), Left Shoulder (12)
  + Head (0)
  + Left Ankle (27), Right Ankle (28)
* The extracted coordinates are stored as **(x, y, z)** tuples for calculations.

**4. Calculations for Fall Detection**

**4.1. Head-Shoulder Positioning**

The primary criterion for detecting a fall is the position of the **nose relative to the shoulder line**:

* If this condition holds, the system classifies it as a fall.
* However, this approach lacks precision in cases where a person is **bending forward, sitting, or crouching**.

**4.2. Ankle Displacement for Motion Detection**

To differentiate between static and dynamic states, the ankle position is tracked across frames:

* If is **significantly large**, movement is detected, meaning the person is unlikely to be in a fallen state.
* The lack of explicit angle calculations makes this approach inferior in detecting actual falls compared to a spinal vector approach.

**5. Visualization and Alerts**

* **Pose Skeleton**: MediaPipe landmarks and connections are drawn on the frame.
* **Text Overlay**: Displays **activity status**.
* **Alert Trigger**: If a fall is detected, the system highlights the person and triggers a warning.

**6. Disadvantages of This Approach**

1. **No Explicit Angle Computation** – Unlike the spine-vector method, this approach does not calculate **spinal inclination**, making it prone to errors.
2. **Higher False Positives** – A person **leaning forward or sitting** may be misclassified as falling.
3. **No Directional Awareness** – The system does not analyse if the person is **falling sideways or lying down intentionally**.
4. **Noise in Landmark Detection** – Fast movement or **poor lighting** may introduce errors in pose tracking.
5. **No Temporal Tracking** – The approach lacks time-based analysis, meaning **slow falls may not be detected effectively**.

**7. Conclusion and Comparison to Superior Methods**

While this method provides a **basic pose-based fall detection system**, it lacks the accuracy and robustness of a **spinal vector approach**, which explicitly computes body inclination and orientation. Future improvements could include:

* **Angle-based analysis** using spine inclination for better classification.
* Temporal tracking to differentiate between lying down and falling.
* **Machine learning-based action recognition** to enhance accuracy.

By incorporating these enhancements, a more **reliable and accurate** fall detection system can be developed for industrial environments.

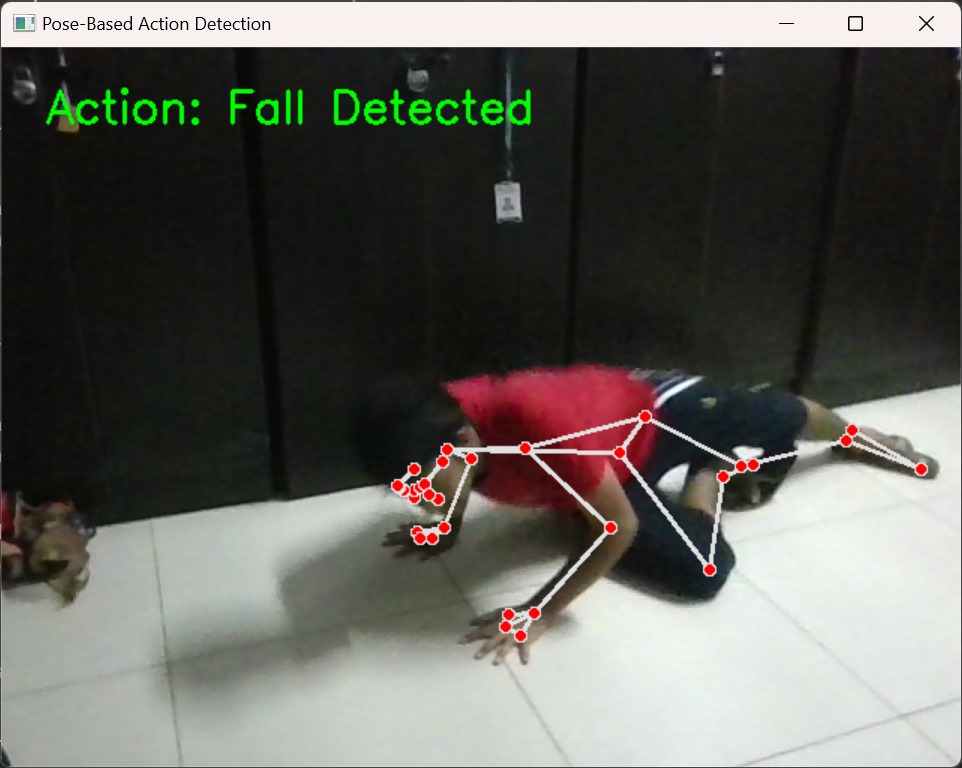
 **8. Output of Implementation:**

Figure 03: Fall Detected

**Methodology: Fall Detection Using Spine Vector Calculation**

**1. Introduction**

This approach leverages MediaPipe Pose to detect and analyze human posture in real-time. Instead of relying on bounding box aspect ratios, this method estimates key joint positions and their relative angles to determine if a worker has fallen. The method computes the spinal inclination angle and checks the relative height of key joints to detect falls accurately.

**2. Model and Dataset**

MediaPipe Pose is used for landmark detection due to its efficiency and robustness.

The model is pre-trained and optimized for real-time human pose estimation.

The system captures and processes frames using OpenCV.

The approach does not require additional dataset training but can be fine-tuned with domain-specific pose correction models.

**3. Frame Processing and Landmark Extraction**

* The system captures video frames and converts them to RGB format for processing.
* MediaPipe extracts 33 key landmarks corresponding to human joints.
* The key joints used in fall detection:
  + Right Shoulder (11), Left Shoulder (12)
  + Right Hip (23), Left Hip (24)
  + Head (0)
* The extracted coordinates are stored as **(x, y, z)** tuples for calculations.

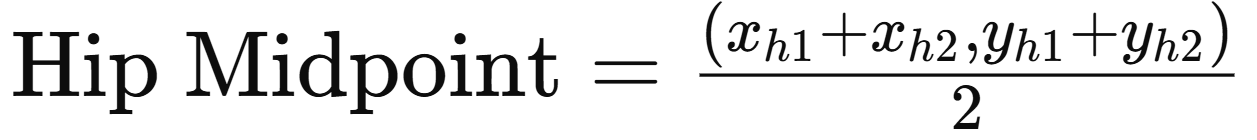
**4. Calculations for Fall Detection**

**4.1. Spinal Inclination Angle**

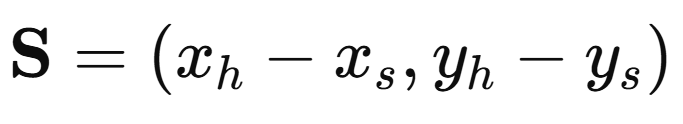
To determine the body’s orientation, we compute the **spinal inclination angle** (θ) between the shoulder-hip vector and the vertical axis.

* Define the midpoint of shoulders and hips:

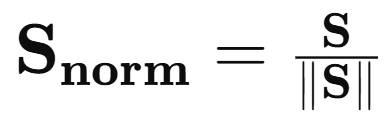




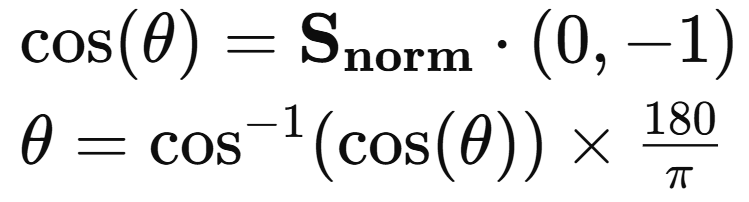
* Compute the spinal vector:



* Normalize the vector:



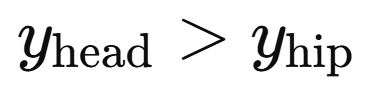
* Calculate the dot product with the **vertical axis** (0, -1):



* If **θ > 70°**, the person is likely falling.

**4.2. Head-Hip Height Comparison**

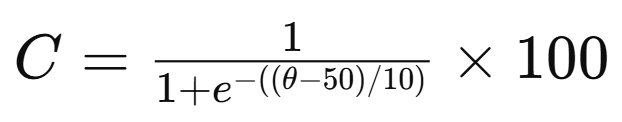
To confirm a fall, we check if the **head is below the hip**:



If true, this further reinforces fall detection.

**4.3. Confidence Calculation (Logistic Function)**

A confidence score is computed using a **logistic function** to normalize fall likelihood between **0-100%**:



* θ = 50° is used as a transition point.
* The function maps angles below 50° to lower confidence values and above 70° to high confidence values.

**5. Visualization and Alerts**

* **Pose Skeleton**: MediaPipe landmarks and connections are drawn on the frame.
* **Text Overlay**: Displays **frame count, activity status, and confidence percentage**.
* **Alert Trigger**: If a fall is detected, the system highlights the person and triggers a warning.
* Future integration: **Automated emergency notifications** to supervisors or **audible alarms**.

**6. Disadvantages of This Approach**

1. **Sensitivity to Camera Angles** – The model assumes a **side-view**; top-down views affect accuracy.
2. **Noise in Landmark Detection** – Fast motion or **poor lighting** can introduce landmark errors.
3. **No Temporal Tracking** – A person lying down for a break might be classified as falling.
4. **Scaling Issues** – Body proportions vary across individuals, requiring adaptive thresholds.
5. **Limited Environmental Context** – Cannot differentiate **intentional sitting/lying** from a fall without additional context.

**7. Implementation Considerations**

* **Real-Time Processing**: The system should maintain at least **30 FPS** for smooth operation.
* **Latency Reduction**: Efficient processing is required for quick detection.
* **Edge Deployment**: Optimized for running on low-power edge devices like **Jetson Nano**.
* **False Positive Handling**: Adaptive thresholds and multi-frame confirmation reduce false alarms.
* **Integration with Wearable Sensors**: Combining **IMU (Inertial Measurement Units)** can improve accuracy.

**8. Conclusion and Future Improvements**

This methodology provides a structured approach for **fall detection using pose estimation**. Future improvements could include:

* **Temporal analysis** using optical flow or tracking over multiple frames.
* **Deep learning-based action recognition** to classify falling vs. lying.
* **Adaptive thresholding** for varying body sizes.
* **Multi-camera integration** for a more holistic assessment.
* **Integration with IoT systems** for real-time alerts and logging.
* **AI-driven anomaly detection** to differentiate between different types of falls and normal activities.

By refining these aspects, fall detection can achieve higher accuracy and robustness in industrial environments.

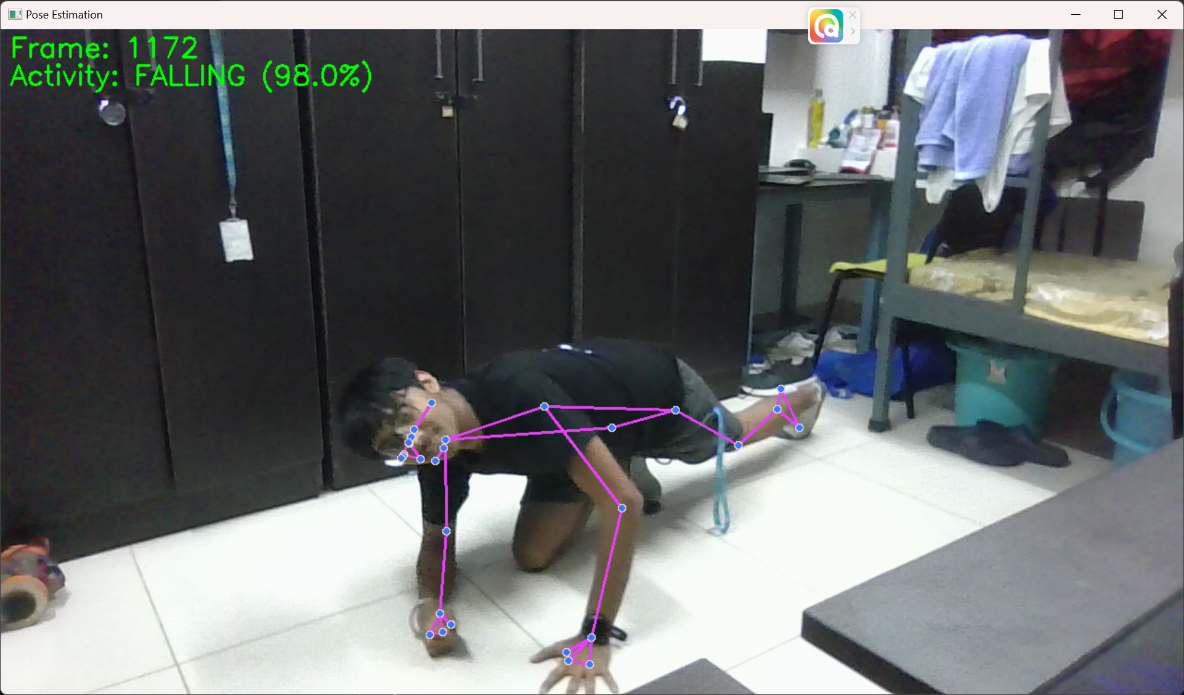
**8. Output of Implementation:**

Figure 04: Fall Detected