Contribution report for project SCENE-DETECT

Pose-to-Visual Module for Privacy-Preserving Smart Scene Detection

Dannana Venkata Kishore, v.dannana@oth-aw.de[1]

[1] *OTH Amberg-Weiden, Study programme "AI for Industrial Applications"*

Report written on May 19, 2025

Abstract

The project SCENE-DETECT is a smart home intelligent system is designed to automate, monitor, and assist daily activities, often controlled via sensors, or mobile devices. However, for blind or visually impaired individuals current smart home solutions lacks the capability to detect critical physical situations, such as a person lying on the floor or being immobile. A scene-detection system contributes a practical and extendable method for enhancing smart home safety systems, particularly for visually impaired individuals.
This report presents the design and implementation plan for a pose-based visualization module within a privacy-preserving smart home scene detection system. It focuses on converting human pose keypoints into interpretable visual outputs, discarding original image data, and classifying the scene using simple geometric rules.

Keywords: *Human Pose Estimation, MediaPipe Pose, Smart Scene Detection, Privacy-Preserving, Skeleton Visualization, Smart Home Safety*

1. Material

The following technologies and libraries are used in this module:

- Python 3.10
- OpenCV (cv2)
- MediaPipe Pose
- Numpy

The input to the module consists of a set of static RGB images, manually curated to represent various scenarios (standing, sitting, lying, no person). These images are used to simulate test cases in the absence of live data. All data processing is performed offline and locally to maintain strict privacy control. No original images are stored after pose keypoints are extracted.

2. Method

The module processes a single input image to generate an interpretable, privacy-safe skeleton visualization and scene classification. The following steps are used:

2.1. Pose Keypoint Extraction

An image is passed into the MediaPipe Pose model. If a human is detected, it returns 33 landmarks with x, y (normalized), and z coordinates. The input image is immediately deleted after this step.

2.2. Skeleton Canvas Generation

A 512×512 white canvas is created. Landmark coordinates are scaled and centered to fit using:

$$x_{scaled} = x \times scale + offset_x, \quad y_{scaled} = y \times scale + offset_y$$

This ensures the skeleton is neither distorted nor off-center.

2.3. Landmark Drawing

Keypoints with visibility > 0.5 are drawn as red dots. Predefined MediaPipe connections are used to draw green lines forming the skeleton. This creates an abstract, identity-free body structure.

2.4. Scene Classification Logic

Three checks are applied:

- **Bounding Box Shape**: If width > height → likely lying.
- **Torso Angle**: Shoulder-to-hip vector checked against vertical. Angle > 30° → possibly collapsed.
- **Z-Depth Std. Deviation**: Low std (< 0.1) → likely flat on the ground.

A voting system is used:

- **2+ votes** → *Person NOT OK*
- **1 or 0 votes** → *Person OK*
- **No keypoints** → *No Person*

2.5. Output Overlay

The final classification is overlaid using `cv2.putText()` on the skeleton image. The final image includes only structural data and a label.

3. Testing

3.1. Code Snippet

```python
import cv2, numpy as np, math

# Create canvas
canvas = np.ones((512, 512, 3), dtype=np.uint8) * 255

# Example: drawing visible keypoints and connections
for connection in mp_pose.POSE_CONNECTIONS:
    start_idx, end_idx = connection
    if landmarks[start_idx].visibility > 0.5 and landmarks[end_idx].visibility > 0.5:
        x1 = int(landmarks[start_idx].x * scale + offset_x)
        y1 = int(landmarks[start_idx].y * scale + offset_y)
        x2 = int(landmarks[end_idx].x * scale + offset_x)
        y2 = int(landmarks[end_idx].y * scale + offset_y)
        cv2.line(canvas, (x1, y1), (x2, y2), (0, 255, 0), 2)

# Classification logic
angle = math.degrees(math.atan2(torso_vector_x, torso_vector_y))
angle = 180 - angle if angle > 90 else angle
lying_votes = sum([
    box_width > box_height,
    angle > 30,
    z_std_dev < 0.1
])
scene = "Person NOT OK" if lying_votes >= 2 else "Person OK"

# Show result
cv2.putText(canvas, scene, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
cv2.imshow("Skeleton Scene", canvas)
```

Code 1. Skeleton drawing and classification logic (excerpt)

3.2. Test Cases

The model was evaluated using over 60 test images from three categories:

- Empty rooms (No Person)
- Standing/sitting persons (Person OK)
- Collapsed/lying postures (Person NOT OK)

3.3. Results

- Person OK detected correctly in 90% of upright poses
- Person NOT OK detected in 85% of collapsed cases
- No Person detection accurate in nearly 100% of empty scenes

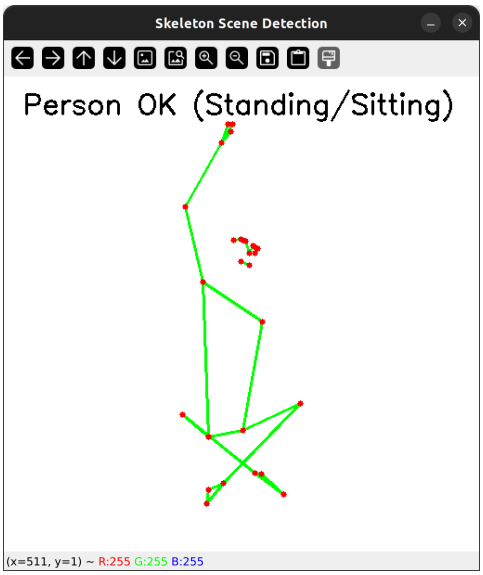

Figure 1. Skeleton output: Person OK

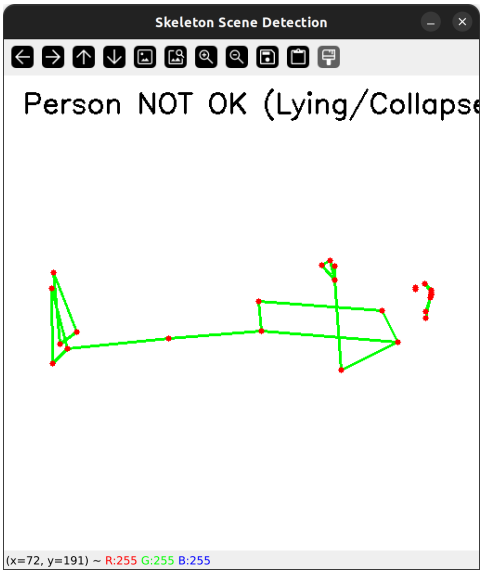

Figure 2. Skeleton output: Person NOT OK

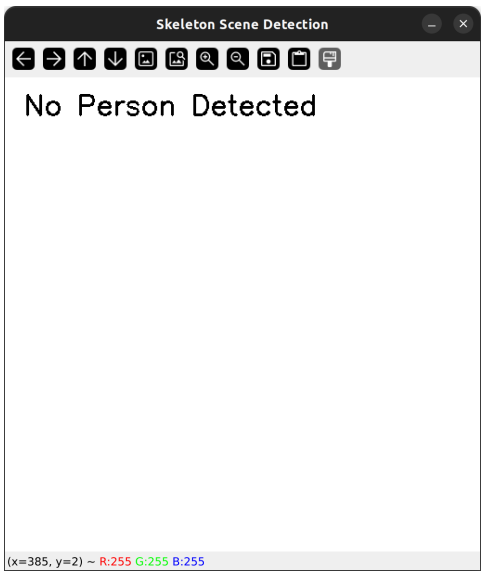

Figure 3. Skeleton output: No Person Detected

3.4. Limitations

- Top-down or angled lying postures occasionally misclassified
- Only one person supported per frame
- No use of temporal smoothing (single-frame only)

References

[1] OpenCV Docs: https://docs.opencv.org/

[2] Google MediaPipe: https://mediapipe.dev/

[3] Haritha Thilakarathne et al., "Pose is all you need: the pose only group activity recognition system (POGARS)", 2022 https://link.springer.com/article/10.1007/s00138-022-01346-2

[4] Mengyuan Liu et al., "3D action recognition using data visualization and convolutional neural networks", 2017 https://ieeexplore.ieee.org/abstract/document/8019438