CCTV Fall Detection System

Rule-Based Multi-Person Analysis

1 Overview

This project implements a **rule-based fall detection system** for CCTV videos. It is designed for indoor surveillance scenarios such as elderly monitoring, security cameras, and public areas.

The system detects and tracks people, analyzes body posture and motion, and determines fall events using a **consensus-based decision logic**. No single rule is allowed to trigger a fall.

2 Key Features

- Multi-person detection and ID-based tracking
- Fall detection using posture, motion, and stability evidence
- Occlusion-aware and partial-visibility handling
- Protection against false positives such as running or jumping
- Fully rule-based (no training required)

3 Project Structure

```
.
        main.py              # Main processing loop
        config.py            # Configuration & thresholds
        utils.py             # Motion, posture, geometry utilities
        tracker.py           # Person and body-part tracking
        fall_analyzer.py     # Fall evidence & consensus logic
        input_videos/        # Input videos
        output_videos/       # Processed output videos
```

4 Fall Detection Logic

Each person is assigned a state:

`STANDING → FALLING → FALLEN`

A fall is detected only if **multiple independent evidences agree over time**.

4.1 Evidence Groups

1. **Posture Evidence**: body angle change, collapse of upright form
2. **Motion Evidence**: vertical velocity or acceleration
3. **Stability Evidence**: loss of control, high variance
4. **Spatial Evidence**: proximity to ground or occlusion near ground

4.2 Consensus Rule

- At least **two evidence groups** must be active
- **Posture or stability** must be one of them
- Evidence must persist across multiple frames

This prevents false positives caused by running, walking, or sudden gestures.

5 Running on Google Colab

5.1 Open Colab

Go to:

`https://colab.research.google.com`

Create a new notebook.

5.2 Enable GPU

In the Colab menu:

```
Runtime  Change runtime type  Hardware accelerator  GPU
```

5.3 Upload Project Files

Upload all `.py` files and the `input_videos/` directory, or clone from GitHub:

```
!git clone <repository-url>
%cd <repository-folder>
```

5.4 Install Dependencies

```
!pip install opencv-python ultralytics numpy
```

5.5 Test Imports

```
print("Testing module imports...")
from config import *
from utils import *
from tracker import new_track, new_body_part_track
from fall_analyzer import analyze_fall_indicators
print("All modules imported successfully!")
```

5.6 Run the System

```
!python main.py
```

Console output will show frame progress and detected fall events.

6 Output

Processed videos with annotations are saved to:

```
output_videos/
```

Detected fall events are drawn on the video and logged in the console.

7 Configuration

All thresholds and parameters can be adjusted in `config.py`, including:

- Number of frames required to confirm a fall
- Ground proximity threshold
- Occlusion tolerance

It is recommended **not to loosen thresholds arbitrarily**, as the system relies on consensus rather than sensitivity.

8 Known Limitations

- Extreme camera shake may affect tracking
- Very low-resolution videos reduce posture accuracy
- Long-term full occlusion may delay detection

These limitations are typical in CCTV systems.

9 Design Philosophy

> *Falls are decided by agreement between multiple independent physical evidences over time, never by a single signal.*

This design prioritizes:

- Low false positives
- Interpretability and debuggability
- Robust behavior in real CCTV environments

10 Future Improvements

- Confidence score instead of binary decision
- Explicit running / jogging classification
- ML-based refinement on top of rules
- Real-time alert integration