Multi-Scenario Detection System - Complete Flow Explanation

System Overview

The Multi-Scenario Detection System is a comprehensive surveillance solution that uses YOLOv8 (You Only Look Once version 8) deep learning model to detect multiple scenarios in real-time from multiple camera feeds. The system supports four detection modes that can be dynamically enabled or disabled:

1. **Person Detection** - Identifies and counts people in the frame
2. **Fall Detection** - Detects when a person falls down
3. **Fight Detection** - Identifies aggressive behavior based on rapid movements
4. **Crowd Detection** - Alerts when too many people are present

Flow Diagram Explanation

1. System Initialization Phase

```
START → Initialize System
```

- **Load YOLO Model**: The YOLOv8 model is loaded either from local storage or downloaded
- **Initialize Flask Server**: Web server starts to serve the dashboard
- **Setup Camera Connections**: Establishes connections to all 4 video sources
- **Create Global Variables**: Initializes counters, event lists, and configuration

2. User Configuration Phase

```
Initialize System → User Selects Detection Scenarios
```

- User accesses the web dashboard
- Checkboxes allow enabling/disabling each detection type
- Configuration is stored in `DETECTION_SCENARIOS` dictionary
- Changes are applied in real-time without system restart

3. Video Processing Loop

```
Video Input → Process Frame
```

- **Multi-threaded Processing**: Each camera runs in its own thread

- **Frame Capture**: Continuously reads frames from video sources
- **Frame Resizing**: Standardizes input to 640x480 for consistent processing
- **YOLO Detection**: Processes frame through the neural network

4. Detection Decision Point

```
Process Frame → Selected Scenarios Enabled?
```

This is a critical decision diamond that routes processing based on user selection:

- **If scenarios are selected**: Routes to appropriate detection modules
- **If no scenarios selected**: Skips processing to save resources

5. Parallel Detection Processing

The system performs multiple detections simultaneously:

Person Detection

- Identifies all people in the frame
- Draws green bounding boxes
- Counts total persons
- Updates global counter

Fall Detection

- Tracks person orientation (height vs width ratio)
- Detects transition from standing to fallen state
- Requires confirmation over multiple frames
- Displays "FALL" label inside red bounding box

Fight Detection

- Analyzes movement history of each person
- Calculates movement velocity and variance
- Detects rapid, erratic movements
- Shows "FIGHT" label in blue bounding box

Crowd Detection

- Counts total people in frame
- Compares against configurable threshold
- Requires sustained crowd for 10+ frames
- Displays warning overlay

6. Tracking and Analysis

```
All Detections → Update Tracking & Movement Analysis
```

- **Person Tracking**: Maintains identity across frames using IoU matching
- **Movement History**: Stores position history for each tracked person
- **State Management**: Updates standing/fallen/fighting states
- **Recovery Detection**: Monitors if fallen person stands back up

7. Visualization

```
Update Tracking → Draw Bounding Boxes & Labels
```

- Draws color-coded bounding boxes:
  - Green: Normal person detection
  - Red (thick): Fall detected
  - Blue: Fight detected
  - Orange: Crowd warning
- Adds labels inside boxes for clarity
- Updates status bar with active scenarios

8. Event Generation

```
Draw Boxes → Detection Threshold Met?
```

Another decision point that determines if an event should be logged:

- **Threshold Check**: Compares detection confidence against configured thresholds
- **Cooldown Verification**: Prevents event spam using cooldown periods
- **Event Creation**: Generates timestamped event with details

9. Dashboard Update

```
Generate Event → Update Dashboard
```

- **Statistics Update**: Increments appropriate counters
- **Event List Update**: Adds new events to the list
- **Real-time Display**: Updates web UI via AJAX polling
- **Alert Generation**: Can trigger external alerts

10. Frame Streaming

```
Update Dashboard → Stream Frame to Web UI
```

- Encodes processed frame as JPEG
- Streams via Flask's multipart response
- Displays in browser using `<img>` tags
- Updates at ~30 FPS

11. Continuous Loop

```
Stream Frame → Loop back to Video Input
```

- Process continues indefinitely
- Each camera thread operates independently
- System runs until manually stopped

Data Flow Architecture

Input Layer

- **4 Video Sources**: Cameras or video files
- **User Configuration**: Detection scenario selections

Processing Layer

- **YOLO Model**: Neural network for object detection
- **Detection Modules**: Specialized algorithms for each scenario
- **Tracking System**: Maintains object persistence

Data Storage Layer

- **Global Counters**: Real-time statistics
- **Event List**: Historical detection log
- **Camera Frames**: Processed video frames

Output Layer

- **Web Dashboard**: Real-time visualization
- **REST API**: Data access endpoints
- **Alert System**: External notifications

Key Design Patterns

1. **Multi-threading**: Each camera processed independently
2. **Observer Pattern**: UI updates based on data changes
3. **State Machine**: Tracks person states (standing, fallen, fighting)
4. **Factory Pattern**: Dynamic creation of person trackers
5. **Singleton Pattern**: Global configuration and counters

Technical Components

Backend (Python)

- **OpenCV**: Video processing and visualization
- **YOLOv8**: Object detection model
- **Flask**: Web server and API
- **NumPy**: Numerical operations

Frontend (JavaScript)

- **AJAX Polling**: Real-time updates
- **Event Handlers**: Checkbox interactions
- **DOM Manipulation**: Dynamic UI updates

Communication

- **REST API**: Configuration updates and data retrieval
- **WebSocket Alternative**: Video streaming via multipart responses
- **JSON**: Data exchange format

This architecture ensures scalable, real-time detection with minimal latency while providing a user-friendly interface for monitoring and configuration.