Detection Method

The detection method relies on the **YOLOv8 (You Only Look Once) model**, specifically the yolov8x.pt variant, which is a pre-trained deep learning model known for its high accuracy and efficiency in object detection tasks. The implementation details are as follows:

- **Model Loading**: The model is loaded using the ultralytics library with the command from ultralytics import YOLO, indicating the use of the YOLOv8 framework.
- **Detection Process**: The video is processed frame by frame using OpenCV's cv2.VideoCapture. For each frame, the model.track() function is called with the parameter persist=True, which ensures consistent tracking of objects across frames by maintaining the state of detected objects.
- **Object Filtering**: The system filters detection results to focus on humans only, using class ID 0, which corresponds to the "person" class in the COCO dataset commonly used with YOLO models. This filtering ensures that only people are tracked and counted, ignoring other detected objects like vehicles or animals.
- **Output Visualization**: Detected people are visualized on the frame with green bounding boxes and their corresponding track IDs, providing a clear visual representation of the detection and tracking process.
- **Additional Features**: Beyond basic detection, the system generates a heatmap to visualize the density of movement, which is blended onto the video frames. This heatmap aids in understanding traffic patterns but is secondary to the IN/OUT counting functionality.

The choice of YOLOv8, particularly the yolov8x.pt variant, suggests a focus on accuracy, as this variant is known for its enhanced performance compared to lighter versions like yolov8n.pt or yolov8s.pt. Research suggests that YOLOv8 is well-suited for real-time applications due to its balance of speed and accuracy, making it appropriate for video processing tasks like this one.

Line Coordinates

The system defines two horizontal lines across the video frame to serve as boundaries for monitoring movement. These lines are calculated based on the dimensions of the video frame, specifically its height (h) and width (w), and are visualized to aid in understanding the counting logic. The details are as follows:

- **Line Definitions**:
  - line_up: Set at 10% of the frame height from the top, calculated as int(h * 0.1). This line acts as the upper boundary for counting downward movements.
  - line_down: Set at 90% of the frame height from the top, calculated as int(h * 0.9). This line acts as the lower boundary for counting upward movements.
  - Both lines span the entire width of the frame, from x-coordinate 0 to x-coordinate w, creating horizontal thresholds across the video.

- **Visualization**:
    - line_up is drawn on the frame in blue ((255, 0, 0)), providing a visual cue for the upper boundary.
    - line_down is drawn in red ((0, 0, 255)), providing a visual cue for the lower boundary.
    - These colors are chosen for contrast and visibility, with blue typically associated with the top and red with the bottom, aligning with common visual conventions.
- **Purpose**: The lines are critical for the IN/OUT counting logic, as they define the thresholds at which movements are counted. The positioning at 10% and 90% of the frame height ensures that most significant movements across the frame are captured, though the exact placement could be adjusted based on specific use cases (e.g., different camera angles or environments).

This setup allows for a clear visual and logical separation of the video frame into regions for counting, with the lines serving as fixed reference points for tracking movement direction.

Logic for Determining IN/OUT Status

The logic for determining whether a detected person is counted as "IN" or "OUT" is based on the movement of their centroid relative to the line_up and line_down coordinates. The system uses tracking to maintain the history of each person's position and applies conditional logic to determine crossings. The details are as follows:

- **Tracking Mechanism**:
    - For each detected person, the system calculates the centroid of their bounding box, defined by the coordinates (cx, cy), where cy is the vertical (y-coordinate) position of the center.
    - The track_memory dictionary is used to store the history of cy positions for each unique track ID, which is assigned by the YOLOv8 tracking module. This history is limited to the last 30 positions (max_track_len = 30) to manage memory and focus on recent movement, ensuring the system remains efficient for real-time processing.
    - The tracking ensures that the same person is consistently identified across frames, even if their appearance changes slightly due to movement or occlusion.
- **Movement Detection**:
    - To determine the direction of movement, the system compares the current cy (current y-position) with the previous cy (prev_y), which is typically the second-to-last position in the track history. This comparison allows the system to infer whether the person is moving upward (decreasing cy) or downward (increasing cy).
- **IN Logic**:
    - A person is counted as "IN" when the following conditions are met:
        - Their previous y-position (prev_y) is above line_up, i.e., prev_y < line_up.
        - Their current y-position (cy) is at or below line_up, i.e., cy >= line_up.

    - This indicates that the person has crossed the line_up moving downward, which is interpreted as entering or moving into a lower region of the frame.
    - To prevent duplicate counting, the system checks the track ID against the counted_in set. If the track ID is not already in counted_in, the in_count is incremented by 1, and the track ID is added to counted_in. This ensures that each person is only counted once for entering, even if they linger near the line or oscillate.
- **OUT Logic**:
    - A person is counted as "OUT" when the following conditions are met:
        - Their previous y-position (prev_y) is below line_down, i.e., prev_y > line_down.
        - Their current y-position (cy) is at or above line_down, i.e., cy <= line_down.
    - This indicates that the person has crossed the line_down moving upward, which is interpreted as exiting or moving out of a lower region of the frame.
    - Similarly, to avoid duplicate counting, the system checks the track ID against the counted_out set. If the track ID is not already in counted_out, the out_count is incremented by 1, and the track ID is added to counted_out. This ensures that each person is only counted once for exiting.
- **Display and Visualization**:
    - The IN and OUT counts are displayed on the video frame using OpenCV's cv2.putText() function. The "IN" count is typically shown in green, and the "OUT" count in red, providing a clear visual distinction. These counts are positioned at the top-left corner of the frame for easy visibility.
    - The visualization also includes the bounding boxes (green) and track IDs for each detected person, as well as the blue and red lines (line_up and line_down), reinforcing the boundaries for counting.
- **Protection Against Duplicates**:
    - The use of counted_in and counted_out sets is crucial for maintaining accuracy, especially in scenarios where a person might linger near the lines or move back and forth. By tracking unique track IDs, the system ensures that each crossing is counted only once, preventing inflation of the counts due to repeated detections.