Verläufiger Code für die Kameraerkennung mit mqtt übertragung

```
Code erstellt/angepasst mit Perplexity/Claude

import cv2
import ssl
import math
import paho.mqtt.client as mqtt
from ultralytics import YOLO


# ------------------ Einstellungen ------------------

MODEL_PATH = r"C:\Cam\yolo26n.pt"
SOURCE = 1

LINE_P1 = (300, 400)
LINE_P2 = (200, 100)
LINE_BUFFER = 30

MIN_MOVE_PIXELS = 5

# ------------------ Logik-Variablen ------------------

Room_count = 0
last_positions = {}
```

```python
person_states = {}
crossed_ids = set()


# ------------------ Hilfsfunktionen ------------------

def point_side_of_line(px, py, x1, y1, x2, y2):
    return (x2 - x1) * (py - y1) - (y2 - y1) * (px - x1)


def get_zone(px, py):
    val = point_side_of_line(px, py, LINE_P1[0], LINE_P1[1], LINE_P2[0], LINE_P2[1
])
    length = math.sqrt((LINE_P2[0]-LINE_P1[0])**2 + (LINE_P2[1]-LINE_P1[1])**2)
    dist = abs(val) / length
    if dist < LINE_BUFFER:
        return "zone"
    return "right" if val > 0 else "left"


def draw_info(frame):
    cv2.line(frame, LINE_P1, LINE_P2, (0, 255, 255), 2)
    cv2.putText(frame, f"Raum: {Room_count}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)


# ------------------ Hauptprogramm ------------------

def main():
    global Room_count, last_positions, person_states, crossed_ids

    model = YOLO(MODEL_PATH)
```

```python
    mqtt_client = mqtt.Client(callback_api_version=mqtt.CallbackAPIVersion.VERSION2
)

    mqtt_client.username_pw_set("testuser", "O#%~54{Kf{-c-7t")

    mqtt_client.tls_set(tls_version=ssl.PROTOCOL_TLS_CLIENT)

    mqtt_client.connect("5ea1e51a4f614745a394b1edd0259a6d.s1.eu.hivemq.cloud", 8883
, 60)

    mqtt_client.loop_start()

    print("Starte Kamera... Drücke 'q' zum Beenden")


    for result in model.track(

        source=SOURCE,

        show=False,

        stream=True,

        persist=True,

        classes=[0],

        conf=0.5,         # ? reduzierter Schwellwert

        verbose=False,

    ):

        frame = result.plot()

        draw_info(frame)


        if result.boxes is not None and len(result.boxes) > 0:

            boxes = result.boxes

            track_ids = boxes.id


            if track_ids is not None:

                active_ids = set()


                for box, tid in zip(boxes.xyxy, track_ids):

                    track_id = int(tid.item())

                    active_ids.add(track_id)
```

```
                    x1, y1, x2, y2 = box.tolist()
                    cx = int((x1 + x2) / 2)
                    cy = int((y1 + y2) / 2)

                    cv2.circle(frame, (cx, cy), 4, (255, 0, 0), -1)
                    cv2.putText(frame, f"ID {track_id}", (cx + 5, cy - 5),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

                    current_zone = get_zone(cx, cy)

                    if track_id not in person_states:
                        if current_zone != "zone":
                            person_states[track_id] = current_zone
                    else:
                        prev_zone = person_states[track_id]

                        if (prev_zone == "left" and current_zone == "right") or \
                           (prev_zone == "right" and current_zone == "left"):

                            if track_id not in crossed_ids:
                                crossed_ids.add(track_id)

                                if current_zone == "right":
                                    Room_count -= 1
                                    if Room_count < 0:
                                        Room_count = 0
                                    print(f"ID {track_id} EXITED, Raum: {Room_count
}")
                                    mqtt_client.publish("raum/personen",
                                                        f"ID{track_id}
,event=EXIT,raum={Room_count}")
```

```
                                else:
                                    Room_count += 1
                                    print(f"ID {track_id} ENTERED, Raum: {
Room_count}")
                                    mqtt_client.publish("raum/personen",
                                                        f"ID{track_id}
,event=ENTER,raum={Room_count}")

                    if current_zone != "zone":
                        person_states[track_id] = current_zone

                last_positions[track_id] = (cx, cy)

            gone_ids = crossed_ids - active_ids
            crossed_ids -= gone_ids
            for gid in (set(person_states.keys()) | set(last_positions.keys
())) - active_ids:
                person_states.pop(gid, None)
                last_positions.pop(gid, None)

        cv2.imshow("YOLO Room Counter", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cv2.destroyAllWindows()
    mqtt_client.loop_stop()
    mqtt_client.disconnect()
    print(f"\nFinale Zählerstände: Raum={Room_count}")


if __name__ == "__main__":
    main()
```

Revision #3
Created 24 March 2026 07:32:19 by Nico
Updated 28 April 2026 06:50:20 by Nico