```
import cv2
import torch
from tqdm.auto import tqdm

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = (
    torch.hub.load("ultralytics/yolov5", "yolov5s", pretrained=True).eval().to(device)
)

model.conf = 0.35


def detect(source_path, num_track_seconds=5):

    cap = cv2.VideoCapture(source_path)
    FPS = cap.get(cv2.CAP_PROP_FPS)
    total_frames = cap.get(cv2.CAP_PROP_FRAME_COUNT)
    print("FPS: ", FPS)
    print("Total Frames: ", total_frames)

    # imageWidth = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    # imageHeight = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # save_filename = source_path.split(".")[0] + "_result.mp4"
    # writer = cv2.VideoWriter(
    #     save_filename,
    #     cv2.VideoWriter_fourcc("m", "p", "4", "v"),
    #     FPS,
    #     (imageWidth, imageHeight),
    # )

    prev_center = None
    not_moving_frame_count = 0
    is_drowning = False
    for frame_num in tqdm(range(int(total_frames))):
        success, frame = cap.read()
        if success:
            with torch.inference_mode():
                results = model(frame)

            xyxys = results.xyxy[0].cpu().numpy()
            for xyxy in xyxys:
                center = ((xyxy[0] + xyxy[2]) // 2, (xyxy[1] + xyxy[3]) // 2)
                # check if the detected object is a person
                if xyxy[-1] == 0 and prev_center is not None:
                    # check for no movement
                    if (
                        abs(prev_center[0] - center[0]) < 20
                        and abs(prev_center[1] - center[1]) < 20
                    ):
                        not_moving_frame_count += 1

                prev_center = center

                bbox, conf, class_id = xyxy[:4].astype(int), xyxy[4] * 100, xyxy[5]

                if not_moving_frame_count >= (num_track_seconds * FPS):
```

```
            color = (0, 0, 255)
            frame = cv2.putText(
                frame,
                "Drowning: Yes",
                (80, 50),
                cv2.FONT_HERSHEY_DUPLEX,
                1,
                color,
                2,
                cv2.LINE_AA,
            )
            is_drowning = True

        else:
            color = (0, 255, 0)
            frame = cv2.putText(
                frame,
                "Drowning: No",
                (80, 50),
                cv2.FONT_HERSHEY_DUPLEX,
                1,
                color,
                2,
                cv2.LINE_AA,
            )
        out_frame = cv2.rectangle(frame, bbox[:2], bbox[2:], color, 2)
        out_frame = cv2.putText(
            out_frame,
            f"conf: {conf:.2f}",
            bbox[:2],
            cv2.FONT_HERSHEY_DUPLEX,
            0.6,
            color,
            2,
            cv2.LINE_AA,
        )
        center_pt = list(map(int, center))
        out_frame = cv2.circle(out_frame, center_pt, 3, color, -1)

        ret, buffer = cv2.imencode(".jpg", out_frame)
        out_frame = buffer.tobytes()

        yield (
            b"--frame\r\n"
            b"Content-Type: image/jpeg\r\n\r\n" + out_frame + b"\r\n"
        )

#       # writer.write(frame)
#       cv2.imshow("Real-time object detection", out_frame)
#       if is_drowning == True:
#           cap.release()
#           cv2.destroyAllWindows()

#       # press "Q" to stop
#       if cv2.waitKey(1) & 0xFF == ord("q"):
#           break
```

```
    # # release resources
    # cap.release()
    # cv2.destroyAllWindows()


if __name__ == "__main__":
    detect("swim.mp4")
    detect("standby.mp4")
```