

```

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")
```