

参赛队员姓名：戚弘逸

中学：上海星河湾双语学校

省份：上海市

国家/地区：中国

指导教师1姓名：杨旭波

指导教师1单位：上海交通大学

指导教师2姓名：卢曦

指导教师2单位：复旦大学

论文题目：A Swimmer Following Vehicle Based on State
Detection and Cross-Frame Modeling

A Swimmer Following Vehicle Based on State Detection and Cross-Frame Modeling

Hongyi Qi

mineedmund@163.com

Shanghai Starriver Bilingual School

Abstract

Drowning is one of the most important causes of unnatural death of children, and it often occurs in the scene of human negligence. The current fixed-camera-based solutions introduce issues such as target occlusion and difficulty in tracking distant small targets. In order to provide better personalized care for parents or pool supervisors, this paper proposes a new swimmer-following vehicle to dynamically track swimmers' behavior and establish a cross-frame dynamic risk behavior model from the input video. (1) This paper builds a multi-scene swimming pool behavior dataset, including indoor/outdoor, day/night and other scenes. Furthermore, this paper uses AIGC (AI-Generated Content) data augmentation to further enhance the generalization capability and protect the privacy of swimmer. (2) In order to provide better personalized care, this paper proposes a new swimmer following vehicle to dynamically track swimmers' behavior and establish a cross-frame dynamic risk behavior model from the input video. The vanishing point-based detection optimization to improve the likelihood of object detection during the Non-Maximum Suppression (NMS) process. Moreover, this paper uses Mosaic processing to prevent the leakage of swimmers' privacy in public pools. (3) To address the issue of limited computational resources on the vehicle, this paper incorporates components such as EIOU loss function, ACON-C activation function, and Ghostnet architecture to optimize the network. Furthermore, we propose two deployment solutions: local and distributed deployment. The distributed deployment offloads core algorithms such as state detection and modeling to an edge server with enhanced computing resources. The experiments indicate that through distributed deployment, the performance of state detection and modeling can reach up to about 30 FPS. This vehicle can be used by parents for remote monitoring of their children, or it can be temporarily rented at public swimming pools to assist parents seeking personalized care and support pool supervisors in issuing early warning alerts. We hope this vehicle contributes to creating a safe swimming environment for children.

Keywords: Swimmer-Following Vehicle, Cross-Frame Modeling, Distributed Deployment,

Drowning Warning

Table of Contents

Abstract.....	2
1. Background and Motivation.....	4
1.1 Motivation.....	4
1.2 Background & Related Works.....	5
1.3 Our Contribution.....	8
2. Design and Implementation.....	10
2.1 Design Principles.....	10
2.2 Overall Architecture.....	11
2.3 Customized Swimming Pool Dataset.....	12
2.4 Lightweight YOLO Model.....	14
2.5 Cross-Frame Modeling.....	17
2.6 Swimmer Following.....	22
2.7 Mosaic-based privacy protection.....	25
3. Evaluation.....	27
3.1 Edge Server & Vehicle Configuration.....	27
3.2 Lightweight YOLO Model.....	27
3.3 Local vs. Distributed Deployment.....	29
3.4 Cross-Frame Modeling.....	30
4. Limitations & Discussions.....	33
5. Conclusion.....	33
References.....	35

1. Background and Motivation

1.1 Motivation

According to the statistics from the Health Commission and the Ministry of Public Security, 57,000 people drown in China every year, among which children account for 56%, equivalent to 88 children are killed by drowning every day¹. As one of the main summer sports for teenagers, swimming has a wide range of participation in both rural and urban areas. At the same time, with the improvement of Chinese economic level, in addition to the professional indoor swimming pool, many communities have also provided outdoor public swimming pool. However, due to human negligence or poor management, swimming pool drowning accidents occur from time to time.

According to the "Review Report on the Injury status of Adolescents and Children in China" of the China Center for Disease Control and Prevention, drowning has replaced meningitis and HIV to become the primary cause of abnormal death of children in China². According to media reports, a 7-year-old child drowned in a swimming pool in Leshan, Sichuan province, on July 7, 2022³ (Figure 1). From this report, "There were a lot of children swimming in the pool at the time. The boy drowned in the middle of the pool and the depth of the water is about 150cm. There were six to eight lifeguards at the scene, but it seems that no one saw the child struggling to call for help". The tragedy could have been avoided if someone had seen and helped. However, human negligence often occurs; and the very short time that takes to drown (usually four to six minutes to cause death) has become a major hazard in pool safety. At the same time, other dangerous behaviors in the pool, such as running in the pool, diving from high places, etc., are also prone to various non-fatal accidents, which cannot be ignored.



Figure 1. A 7-year-old Child Drowned in a Swimming Pool

¹ <https://new.qq.com/rain/a/20210825A07YZW00>

² https://www.thepaper.cn/newsDetail_forward_19201322

³ <https://www.163.com/dy/article/HBROISBU055206NO.html>

Therefore, if the artificial intelligence assisted personalized care method can be adopted, through state detection and tracking, and modeling the state (such as staying underwater for a long time), dangerous behavior can be found in time. An early warning can be issued, and drowning or other accidents caused by the negligence of pool lifeguards or parents can be reduced, which is helpful to improve pool safety. In summary, this paper explores a new swimmer following vehicle to dynamically track swimmers' behavior and establish a cross-frame dynamic risk behavior model, which is helpful to provide a safer swimming environment for teenagers.

1.2 Background & Related Works

At present, there are three categories of related work in swimming pool detection, modeling and alarm: intelligent life saving system based on underwater camera [1, 2, 17, 18, 21, 22, 23], drowning alarm based on wearable devices [3, 4], and drowning detection methods based on deep learning [5, 19, 20, 22] (as shown in Table 1).

Table 1. Comparison of Related Works

Research Methods	Categories	Deployment Cost	Maintenance Cost	Accuracy	Capability
Underwater Camera	Underwater Life-Saving [1], Life-Saving Monitoring [2], Underwater Surveillance [17,18, 21, 23]	Not popular at present	Maintenance of underwater equipment is difficult	Medium	Mainly drowning detection
Wearable Devices	ZigBee[3], Arm anti-drowning [4]	High	High regular maintenance	High	Drowning and Sudden Disease Detection
Image Detection	Deep learning (Foreground detection [5], Swimmer detection [19], Swimmer Counting [20], 5G network [22])	Low	Low	Medium	Mainly drowning detection

At present, underwater cameras are used in swimming pools (used in places such as the National Aquatics Center). The patent of Beijing Dolphin Light Wave Technology Development Co., Ltd. proposes an anti-drowning warning system based on underwater cameras [1]. Through images and data of swimmers, based on computer vision technology, this system can identify and detect underwater drowning and send alarm messages within 8 seconds. Roman Vestnikov *et al.* use

neural networks directly to search (detect) the necessary objects on the frame to find people in the water [17]. Keshi Li proposes a swimming pool intelligent assisted drowning detection model based on Computer Feature Pyramid Networks [18]. The computer feature pyramid network is used to extract and detect the image features, and the YOLO principle is used to detect the drowning phenomenon in the water. A swimmer behavior recognition framework (BR-YOLOV4) proposed to detect the swimmer [21], using the spatial position relationship between the location information of the target and swimming/drowning area of swimming pool to further determine the swimmer's behavior. Recently, a drowning detection video system with edge computing is proposed, and it can detect drowning events in swimming pools without any wearable devices [23]. However, the installation cost and maintenance requirements of underwater based cameras are high, and they have not been widely used. This paper intends to use the mobile vehicle, which is convenient to deploy, to assist lifeguards to improve their ability to spot accidents and avoid missing the rescue opportunity.

The second category utilizes wearable devices that necessitate swimmers to wear specialized equipment for detecting drowning signals [3, 4]. These devices may include modules for monitoring swimmers' heart rates through ZigBee wireless positioning and heart rate monitoring, along with an integrated alarm button on the bracelet to initiate alarms proactively. However, there are some deployment and maintenance problems, such as the wearable device is easy to fall off in the water or underwater, or the signal transmission is difficult ("ensure that the bracelet is above the water " [3]), or the battery operation time is short. On the other hand, camera equipment is intended to be used in this paper, which does not require swimmers to wear specific equipment. At the same time, it can not only detect drowning, but also detect dangerous behaviors such as slipping and running, which has the characteristics of wide detection range and extensible warning types.

At present, some artificial intelligence methods based on computer vision have been used to study drowning behavior, such as the Mask R-CNN method [5], but it is limited to single frame target detection and belongs to a pure computer vision classification method. However, "This study found that the spatio-temporal information of drowning detection is important, and we hope to explore more spatio-temporal information in the future for more accurate drowning detection" [5]. Morten B. Jensen *et al.* train two convolutional neural networks, YOLOv2 and Tiny-YOLO, to detect swimmers in low quality video [19]. Moreover, a pool counting system was built using the YOLOv3 deep learning algorithm as an object detection algorithm [20]. In fact, "Accuracy value is influenced by the camera's location against sunlight" [20]. An architecture dedicated to child drowning prevention with 5G network slicing has been presented in a recent study [22]. This research incorporated a thorough evaluation and implementation of three renowned CNN models: ResNet-50, VGG-19, and Inception-v3. These models were employed to identify distracted parents or caregivers and promptly notify them to redirect their attention towards actively supervising their children.

However, the current video-based pool monitoring solutions, whether for indoor or outdoor surveillance, are based on fixed cameras (Figure 2).



Figure 2. Current Swimmer Surveillance

The current monitoring system presents several issues:

1. **Target Occlusion.** Swimmers may be obstructed by other swimmers or objects like lifebuoys, as illustrated in Figure 2b where the swimmer in the lower right corner is obscured by a lifebuoy, making recognition difficult. Complex pool layouts, such as bridges (Figure 3a) and other obstructions, can also pose challenges to identification.
2. **Small Target Detection.** Due to the expansive coverage of the surveillance system, swimmers appear relatively small (Figure 2c), increasing the risk of missed detections.
3. **Other dangerous behaviors.** Apart from drowning, there are other dangerous behaviors in the swimming pool, such as running and diving along the poolside (Figure 3b). These are aspects that current target detection methods lack.
4. **Personalized Care.** Parents desire close observation of swimmers to assess their condition, including checking for proper swimming posture or signs of fatigue. However, parents may not be physically present at the pool and rely solely on large screens or outdoor views (Figure 3c), which may not provide clear visibility.

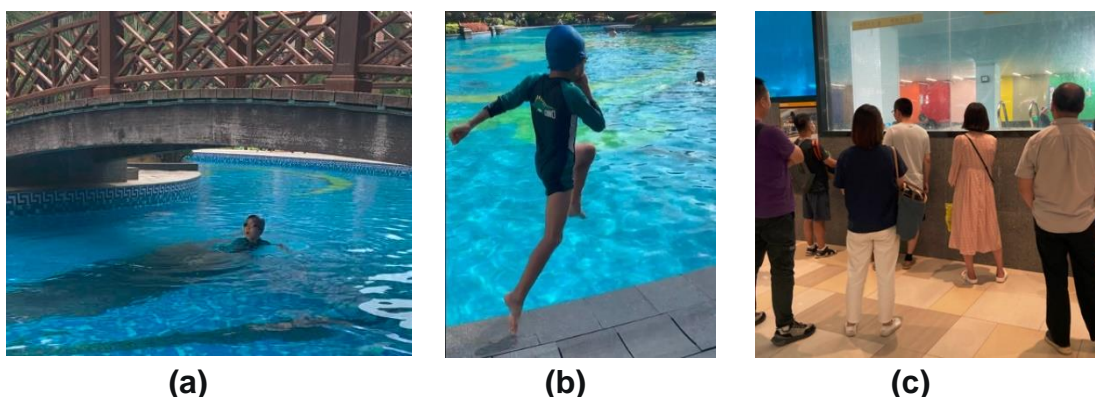


Figure 3. The Limitations of Current Swimmer Surveillance. (a) The view is obstructed by the bridge. (b) Other dangerous behaviors such as jump. (c) Parents outside the pool

⁴ <https://zhuanlan.zhihu.com/p/608107145>

⁵ <http://www.sostech.cn/lm4/>

This paper proposes a new swimmer following vehicle (Figure 4) to dynamically track swimmers' behavior, which provides a richer semantic model with spatio-temporal information, improves the detection accuracy and category, and facilitates the providing of a more intuitive abnormal behavior model. This vehicle can be utilized by parents for remote monitoring of their children, or it can be temporarily rented at public swimming pools to provide assistance to parents in need of personalized care, or to aid pool supervisors in issuing early warning alerts.

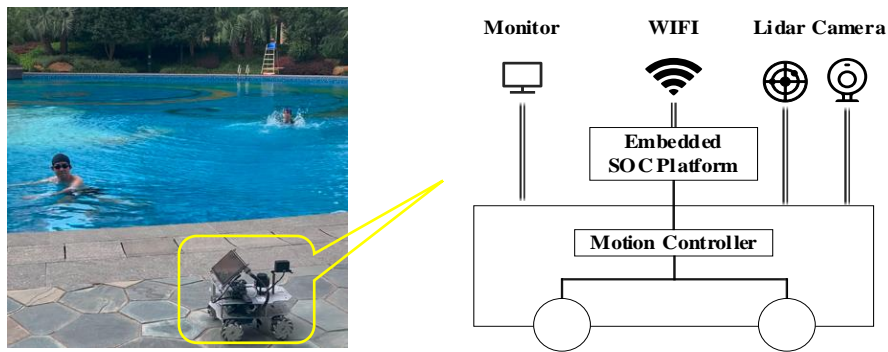


Figure 4. The Swimmer Following Vehicle

1.3 Our Contribution

This paper explores a swimmer following vehicle, serving as an auxiliary monitoring device for parents or pool supervisors. It employs deep learning-based state modeling and autonomous swimmer tracking to offer a novel unmanned pool supervision method. It provides timely alerts for risky behaviors in the pool, contributing to a safer swimming environment for teenagers. The main work is as follows:

- (1) A multi-scene swimming pool behavior dataset is established, including indoor/outdoor, day/night and other scenes. The data are partly from the Internet and also from homemade live videos. Furthermore, to further enhance the generalization capability and protect the privacy of swimmer, we applied data augmentation based on AIGC (AI Generated Content) to the existing dataset.
- (2) The current fixed-camera-based solutions introduce issues such as target occlusion and difficulty in tracking distant small targets. In order to provide better personalized care, this paper proposes a new swimmer following vehicle to dynamically track swimmers' behavior and establish a cross-frame dynamic risk behavior model from the input video. The vanishing point-based detection optimization to improve the likelihood of object detection during the Non-Maximum Suppression (NMS) process. Furthermore, this paper uses Mosaic processing to prevent the leakage of swimmers' privacy in public pools.
- (3) This paper incorporates components such as EIOU loss function, ACON-C activation function, and Ghostnet architecture to optimize the network. Additionally, we introduce three network models, namely Baseline, Ghostnet+,

and BiFPN+CA, and conduct a comparative analysis with the latest YOLOv8 model to evaluate the effectiveness of our optimizations. Specifically, we propose Ghostnet+ as an efficient object detection network model for state detection.

- (4) Due to the limited computing power and energy constraints of the vehicle, this paper proposes a distributed deployment to offload core algorithms such as state detection and modeling to an edge server with more abundant computing resources. The experiments show that the performance of state detection and modeling can reach up to about 30 FPS.

2. Design and Implementation

2.1 Design Principles

In our design process, based on the actual scenario of swimmer following vehicle, we consider the following design principles:

1. **Cross-Frame Modeling.** We want to dynamically track the swimmer's behavior and establish a cross-frame dynamic risk behavior model from the input video, which provides a richer semantic model with spatio-temporal information. We not only track the ID of the swimmers but also perform cross-frame tracking of this ID's state, enhancing recognition accuracy based on their movement direction and appearance characteristics.
2. **Distributed Deployment.** Due to the limited computing power and energy constraints of the vehicle, we are considering to offload certain tasks such as target tracking and state modeling to an edge server with more abundant computing resources. The vehicle will primarily serve as a perception device (camera) and motion controller (wheel movement). This approach aims to enhance performance and make it suitable for embedded devices with limited computational capabilities, such as the Raspberry Pi platform. We also offer local deployment by deploying both tracking and modeling on the vehicle (such as Nvidia Jetson Nano).
3. **Lightweight Network Model.** We found that on the Raspberry Pi, object detection alone takes approximately 2 seconds, which does not meet our requirements. Therefore, on computation-limited vehicle, it is necessary to use lightweight neural networks while ensuring a certain level of accuracy. Additionally, the recently popular Transformer-based neural networks have relatively high computational demands and are less suitable for the vehicle at the present time.
4. **Privacy Protection.** Since swimming pools are mostly public spaces, data privacy of swimmers is a critical concern. Our approach involves pixelating the images of all swimmers unless users specifically request to view images of their own children swimming, thus preventing privacy breaches. During the monitoring process, no data will be stored. Furthermore, in dataset curation, we will exclude images that allow clear identification of swimmers and incorporate images generated by AIGC (AI-Generated Content, i.e., Midjourney⁶) to avoid compromising the privacy of the swimmers.

⁶ <https://www.midjourney.com/>

2.2 Overall Architecture

The overall system architecture is depicted in Figure 5, divided into two components: offline processing and online processing. The offline processing component consists of three parts: the trained lightweight model of the state detection network, the model of swimming pool risk behaviors, and the planned vehicle movement path based on Lidar Mapping. The online processing component operates in a loop, taking input from the pool camera video, and then processing it through three stages: ① State Detection, ② Cross-Frame Modeling, and ③ Swimmer Following. It detects and locates the state and position of the swimmer. Then, the state of the swimmer was dynamically tracked across frames. We then model the state of the swimmer (e.g. staying underwater for a long time); Combined with environmental information, such as weather forecasts, dangerous behaviors are analyzed to alert pool managers or parents. At last, it sends motion instructions to control the vehicle's tracking of swimmers.

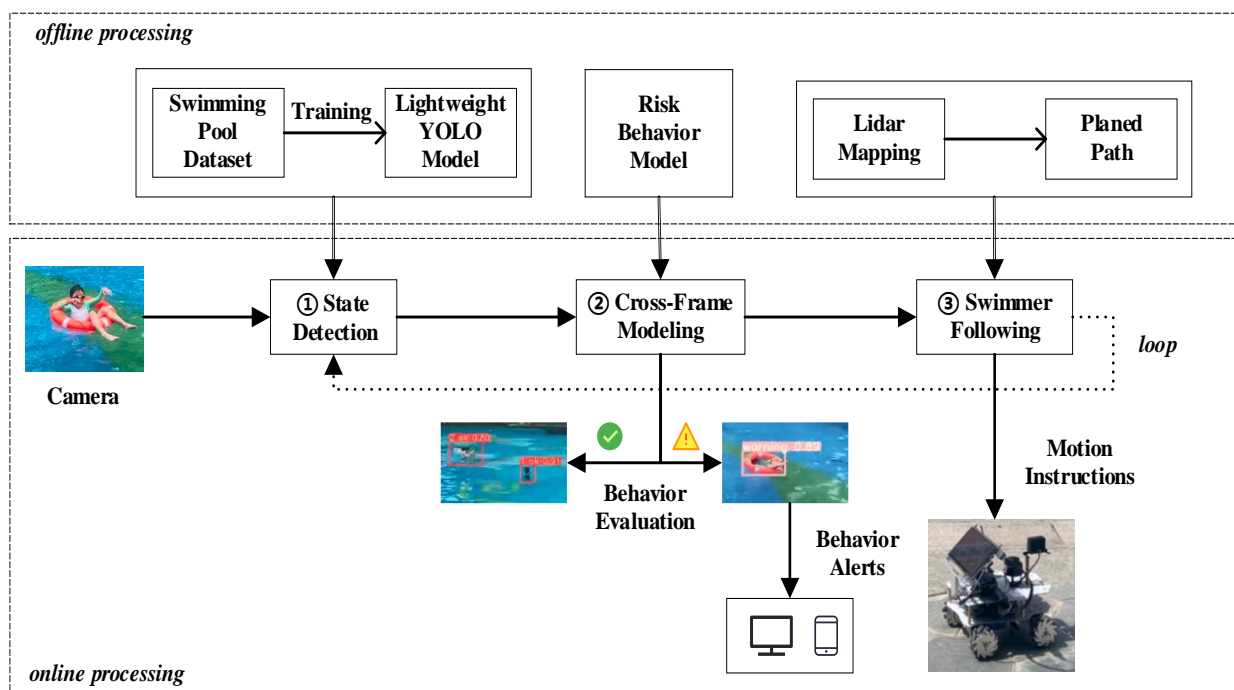


Figure 5. Overall Architecture

Two deployment approaches are used. The first is local deployment, where the core algorithm is deployed on the vehicle itself (Figure 6a). The second approach is distributed deployment, where the vehicle primarily sends camera video to the edge server (Figure 6b). The edge server runs the core algorithm and sends motion instructions to control the vehicle's movement. We will evaluate both deployment methods in Section 3.

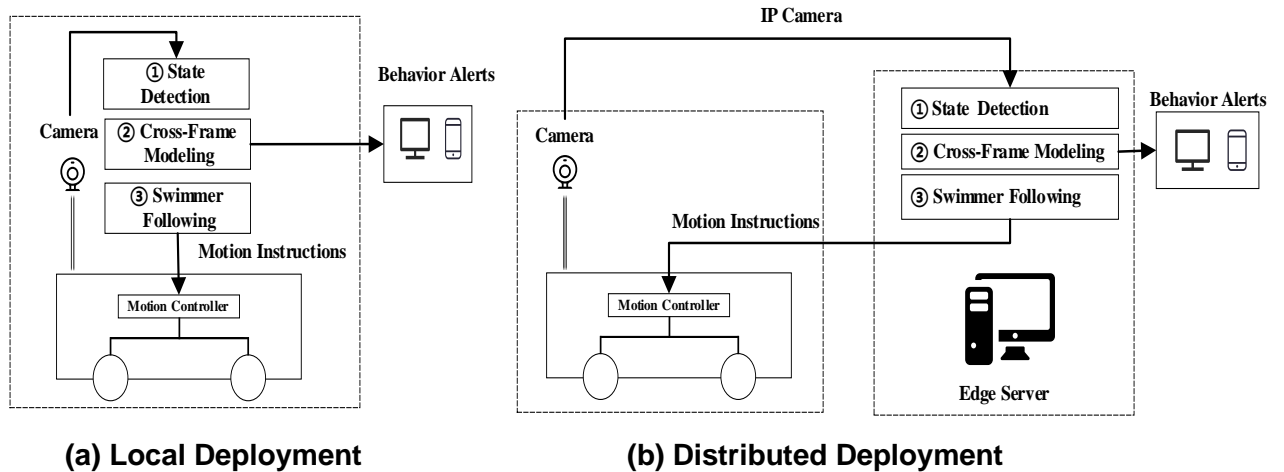


Figure 6. Local and Distributed Deployment

2.3 Customized Swimming Pool Dataset

There is no publicly available dataset of swimming pool and dangerous behaviors, and we produced the dataset for this project by taking and annotating photo and video data from the web search. In order to have a wide range of adaptability, we collect data from typical scenes, including indoor/outdoor, day/night scenes, and also include multiple resolutions and different formats of pictures and videos.

For continuous videos, the number of converted pictures is relatively large, so we only select pictures in typical states to maximize the coverage of data. Considering that the supervision of outdoor swimming pools is the weak link (there is a drowning accident in an outdoor swimming pool in a neighborhood around), we focus on the collection of outdoor live data (accounting for more than half of the dataset). After data collation and annotation, the current dataset consists of 1481 images in the training set and 527 images in the validation set, with a total of 2008 labeled typical scenes. To further enhance the generalization capability of the swimming pool dataset and protect the privacy of swimmers' data, we applied data augmentation based on AIGC (by Midjourney) to the existing dataset. This resulted in an addition of 76 images to the training set and 30 images to the test set.

The current distribution of the dataset is as follows:

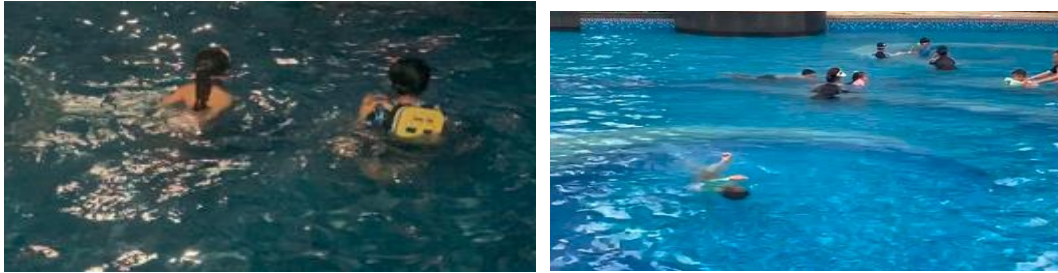
Table 2. Dataset Distribution

Category	Train Set	Test Set
Outdoor (live video)	720	251
Indoor (live video)	450	156
Outdoor (from Internet)	145	55
Indoor (from Internet)	166	65
AIGC (by Midjourney)	76	30



(a) Outdoor (from Internet)

(b) Indoor (from Internet)

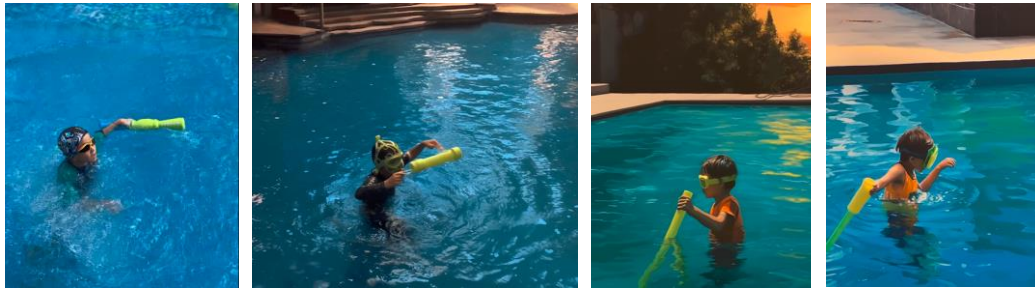


(c) Night scene (live video)

(d) Outdoor (live video)

Figure 7. Multi-scenario pool behavior dataset

We used Midjourney to generate images based on original picture (Figure 8a), creating variations in weather, scenarios, and resolutions (Figure 8b, c, d), thus expanding the dataset.



(a)

(b)

(c)

(d)

Figure 8. Data Augmentation based on AIGC. (a) is original, and (b,c,d) are generated by Midjourney

At present, the dataset will be labeled according to four typical states, namely "ok, underwater, jump and run" (Figure 9). Considering the special scene of the pool, staying underwater for a short time is a normal behavior that often occurs, and staying underwater for a long time is a possible dangerous behavior. Running in the pool is also a dangerous behavior, which also needs to be paid attention to. These states can be further expanded as needed in the future. This section is mainly about marking states, and the dynamic behavior tracking and modeling between states are detailed in Section 2.5. Note that compared to the current traditional object detection (Figure 2b), which only identifies swimmer as "people", we not only identify swimmers here, but also identify the state of swimmers.

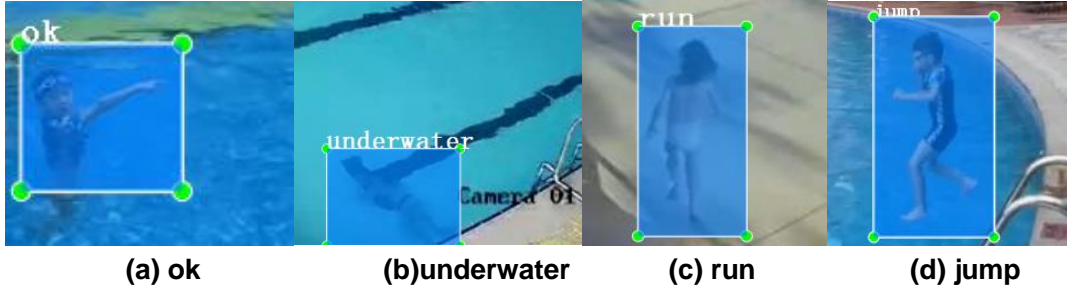


Figure 9. Dataset Labeling

2.4 Lightweight YOLO Model

There are many object detection algorithms based on deep learning. After investigation, we use the well-established YOLO [6] network model as the foundational method, given its maturity, superior generalization capabilities, and active community support. Specifically, we choose YOLOv5 as the state detection and use it as a pre-trained model. Aiming at the requirements of lightweight scene of pool detection, the following optimization is carried out, and compared with the recent YOLOv8⁷.

1. Loss and activation function improvements

YOLOv5 uses CIoU loss function [7], which is defined as follows:

$$L_{CIoU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \quad (1)$$

Where IOU is the traditional Intersection over Union, which is expressed as the ratio of the intersection and union of the "predicted" and "true" bounding boxes. $(\rho^2(b, b^{gt}))/c^2$ is the square of the Euclidean distance between the predicted box b and the true target box b^{gt} divided by the square of the diagonal length c of the minimum bounding box covered by the predicted and target boxes.

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2 \quad (2)$$

$$\alpha = \frac{v}{(1-IOU)+v} \quad (3)$$

Then αv measures the difference in aspect ratio. Reference [8] points out that v in CIoU formula reflects the difference of aspect ratio, rather than the real difference of width and height with their confidence respectively. Once the aspect ratios of the predicted and target boxes scale linearly, the added loss term has no effect. To this end, the EIOU loss function [8] is proposed, which is calculated as follows:

$$L_{EIOU} = L_{IOU} + L_{dis} + L_{asp} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{(w^c)^2 + (h^c)^2} + \frac{\rho^2(w, w^{gt})}{(w^c)^2} + \frac{\rho^2(h, h^{gt})}{(h^c)^2} \quad (4)$$

⁷ <https://github.com/ultralytics/ultralytics>

w^c and h^c are the width and height of the minimum bounding box covered by the predicted and target boxes, respectively. EIOU is divided into three types of losses, the overlap loss LIOU of predicted and target true boxes, the center distance loss L_{dis} of predicted and target boxes, and the width and high loss L_{asp} of predicted and target boxes. Therefore, the EIOU loss directly minimizes the difference between the width and height of the target box and the anchor box, which will have faster convergence speed and better localization results.

ReLU activation function is a commonly used neural network activation function. Recently, Swish activation function (which can be seen as a smooth approximation of ReLU) found by network architecture search technology is also good. Literature [9] has studied this function and proposed a new activation function Family ACON Family. We can explicitly optimize parameter switching between nonlinear (activated) and linear (inactive), and outperform ReLU and Swish activation functions on tasks such as classification and detection. In particular:

$$ACON - C: (p_1 - p_2)x * \sigma(\beta(p_1 - p_2)x) + p_2x \quad (5)$$

Where p_1 and p_2 are the upper and lower limits of the responsible control function, which can be obtained by learning. The parameter β , which dynamically controls the linear/non-linear activation function, is learned by a small convolutional network. σ is the Sigmoid function. YOLOv5-6.1 source code has added the ACON-C activation function, this project will replace the original Swish activation function with this activation function.

2. Lightweight Network Improvements

YOLOv5 has made many improvements in network lightweight and provides a variety of lightweight models. This paper intends to adopt Ghostnet [10] proposed by Huawei Noah's Ark Laboratory as a lightweight backbone network.

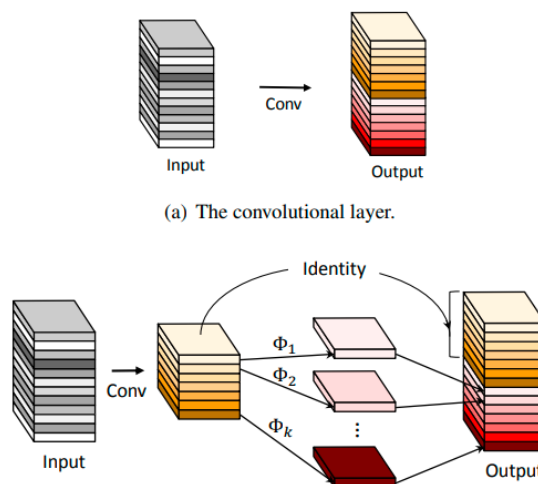


Figure 10. Ghostnet Network Architecture [10]

Based on MobileNetV3, the Ghostnet network divides the traditional convolution operation into two parts. The convolution operation with a small number of convolution kernels is used as the first part, and then the *concat* operation is performed with the second part of the Identity map (see Figure 10) and the

lightweight per-channel convolution operation. The computational complexity of the network is greatly reduced while the accuracy is not decreased. This paper will analyze the impact of Ghostnet on the amount of network computation and evaluate whether it is suitable for pool detection.

3. Feature Fusion and Coordinate Attention Mechanism

YOLOv3 uses feature fusion and multi-scale prediction to effectively optimize the detection of small target objects. In the swimming pool scene, many swimmers' targets are small and most of them are underwater. Therefore, multi-scale feature fusion is an effective method for this scene. In this project, BiFPN (Bidirectional feature pyramid network) [11] is used to realize simple and lightweight multi-scale feature fusion. The traditional feature fusion method does not distinguish the input features, while BiFPN can learn the weight of each feature, which can detect small objects more effectively. The feature map obtained combines the features of the current layer and the upper and lower layers, a total of three layers (see Figure 11). In this project, it is added to the detection network to evaluate the impact on the detection effect of small objects.

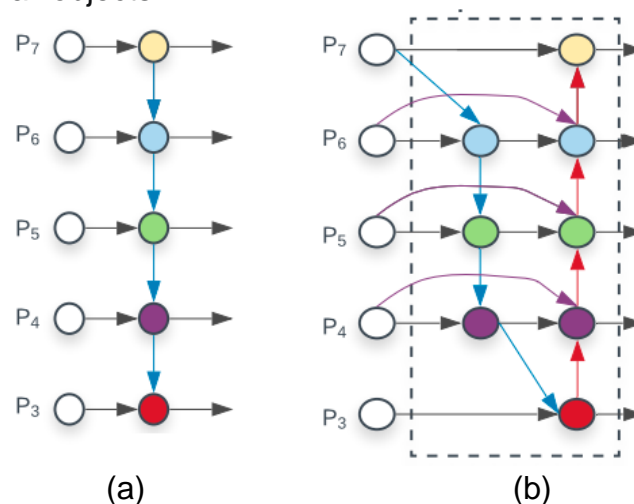


Figure 11. (a) traditional feature pyramid; (b) BiFPN (current layer and two layers above and below) [11]

Attention mechanism is widely used in deep learning, so we try to introduce the spatial Coordinate Attention mechanism (CA) [12] to explore the application of attention mechanism in the pool scene. This method uses two one-dimensional global pooling operations to aggregate the input features along vertical and horizontal directions into two independent direction-aware feature maps, and then encodes the two feature maps with embedded direction-specific information into two attention maps, which capture the long-distance dependence of the input feature maps along a spatial direction respectively. However, in the pool scenario, the spatial position of the swimmer is a continuous state switch, so we try to introduce this mechanism to evaluate whether it is suitable for the pool scenario.

YOLOv8 is further improved on the classic YOLOv5. On January 10, 2023, YOLOv8 was officially open sourced. YOLOv8 is an anchor-free model, which directly predicts the center of the object rather than the offset of the known anchor

box, reducing the number of box predictions and accelerating the processing of NMS. Backbone is also improved, and the C3 module is replaced by the C2f module to further realize lightweight. It improves the overall speed and accuracy on the COCO benchmark. In this paper, YOLOv8 will be compared with the methods we mentioned earlier to evaluate which is more suitable for pool detection.

2.5 Cross-Frame Modeling

With object detection, we obtain the state and coordinates of the swimmer, and the corresponding Confidence. It lays a good foundation for further target state tracking and modeling. We also found that if the object detection is inaccurate, it will affect the object tracking or even lose the object. At present, the commonly used object tracking algorithms include SORT (Simple Online and Realtime Tracking) [13] and improved DeepSORT [14] based on deep learning. In this paper, the DeepSORT-based algorithm is used to track the object state⁸. The SORT algorithm uses Kalman Filter [15] to predict the state of the object in the previous frame in the current frame. For the Mahalanobis distance based on appearance information, it calculates the cost matrix of the tracks of swimmers and the detections of the detections list. Hungarian algorithm [16] is used to match the predicted tracks and detections in the current frame (cascade matching and IOU matching), and the information of the matched pairs and unmatched pairs of swimmers in the current frame is obtained. For the matched swimmers, the mean vector and covariance matrix of the Kalman filter need to be updated.

The main improvement of DeepSORT for SORT is that it borrows the deep learning model in the field of ReID (Re-identification) to extract appearance features (each swimmer extracts a one-dimensional feature vector with a total length of 512 bits), and calculates the similarity through the cosine distance of the feature vectors of two swimmers before and after frames. The similarity results are applied to the Hungarian algorithm used to match swimmers.

Therefore, in the swimming pool scene, due to occlusion and diving, there will be situations where the swimmer does not appear in the current frame, so it is necessary to introduce a set of tracking algorithms that can handle occlusion and loss situations. Specifically, the swimmer in each frame of the monitoring video has obtained the state, position and confidence through the aforementioned object detection, and the feature vector extracted by the Re-ID module is used to determine whether the swimmer in the subsequent frame is the same person. If it is the same person, the corresponding ID does not change. If it is not the same person, if it appears in the preceding frame, it is marked with the ID of the occurrence, otherwise, the swimmer is given a new ID.

⁸ <https://github.com/dyhBUPT/StrongSORT>



Figure 12. Swimmer tracking based on DeepSORT algorithm

With the state of the swimmer and the corresponding ID, the state model can be established to collect the pool risk behavior (such as underwater for a long time, running, etc.), and the risk behavior model can be established according to the dynamic object tracking. According to the behavior model, the swimmer can be tracked in real time and the risk behavior can be warned.

For pool scenes, we introduce vanishing point-based detection optimization to enhance swimmer detection probability. Pools typically exhibit regular shapes, making vanishing point estimation feasible using OpenCV-based detection algorithms. As shown in Figure 13 below, the red dot represents the computed vanishing point.



Figure 13. The vanishing point (red dot) computed by OpenCV

In order to compute the vanishing point, we use *cv2.GaussianBlur* and *cv2.Canny* to detect edges in an image (Figure 14b). Then *cv2.HoughLinesP* is used to get the lines of this image. At last, we find and sort the intersection points of these lines to locate the vanishing point (Figure 14c).

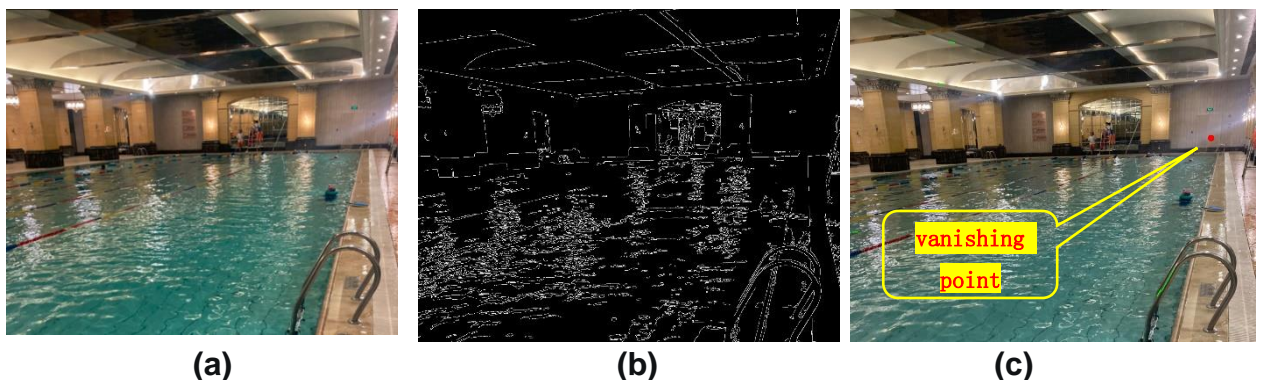


Figure 14. The computation steps for the vanishing point

Now, we pass *vanishingpoint* to the function of *non_max_suppression* to adjust the confidence of detected target, according to the follow formula:

$$\text{distance} = \sqrt{(x - vx)^2 + (y - vy)^2} \quad (6)$$

$$\text{dist_adj} = \text{torch.clamp} \left(1 + 0.2 * \left(1 - \frac{\text{distance}}{\text{width}} \right), \text{min} = 1.0, \text{max} = 1.2 \right) \quad (7)$$

$$\text{confidence} = \text{confidence} * \text{dist_adj} \quad (8)$$

$$\text{iou}_{\text{thres}} = \text{iou}_{\text{thres}} * \text{dist_adj} \quad (9)$$

First, we compute the distance between the current swimmer (x, y) and the vanishing point (vx, vy) by Formula (6). Then we compute the confidence adjust factor *dist_adj* by Formula (7). This formula can be replaced by other computing method. Then we use this factor to adjust the every object's confidence by Formula (8) and the *iou_thres* during during the Non-Maximum Suppression (NMS) process. Figure 15 shows the result of this adjustment. Figure 15a show the confidence without vanishing point. The confidence of the right swimmer is 0.85, but when adjusted as shown in Figure 15b, it increases to 0.92. The swimmer on the left is relatively far from the vanishing point, so there is a limited improvement in confidence.

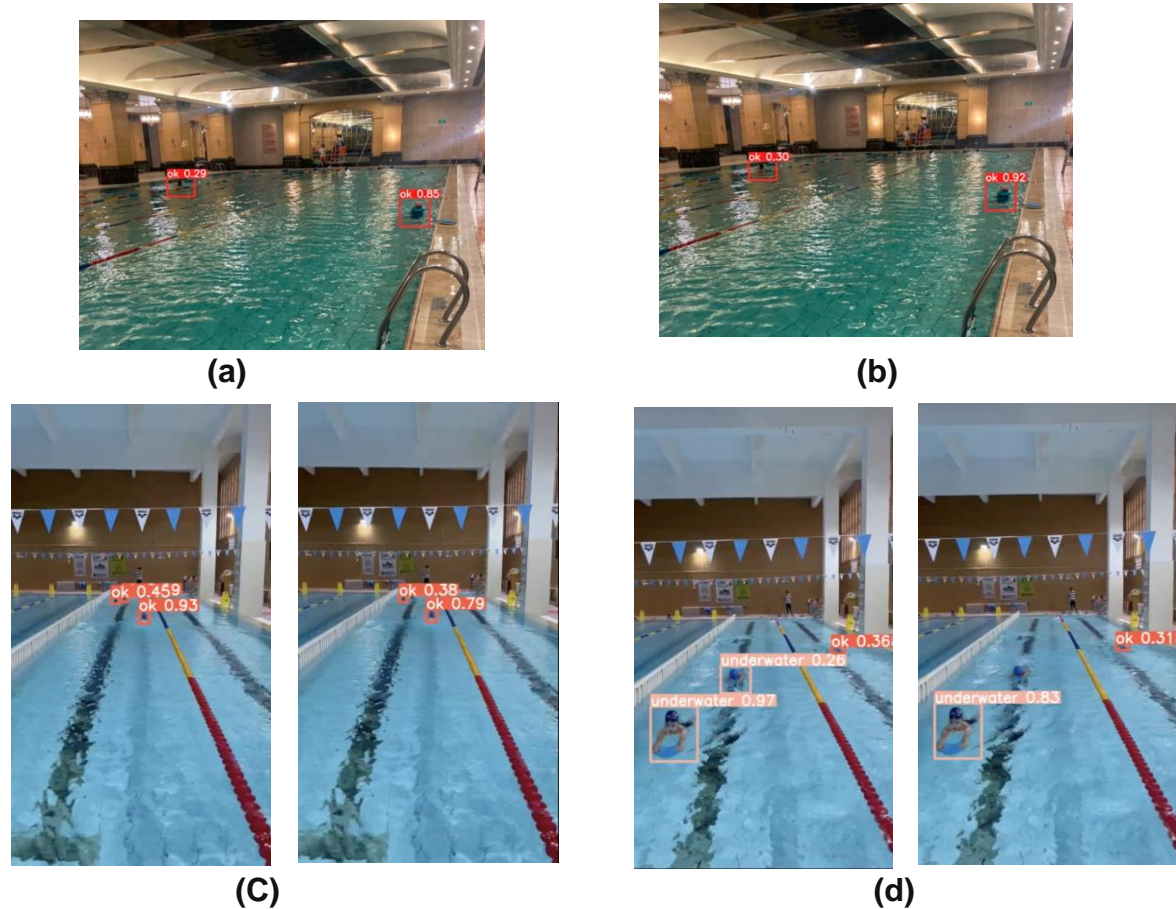


Figure 15. The difference of detection of without/with vanishing point

In Figure 15c, with the optimization of vanishing point, we can identify 3 swimmers (without this optimization, only 2 swimmers) in the case of crowded

people. Also, in Figure 15d, this optimization detect the swimmer ignored by the original version. Hence, based on the introduction of the vanishing point, we have increased the confidence in swimmer detection, thereby enhancing the likelihood of object detection during the NMS process

It mainly tracks the status of swimmers, and sends "SOS" information and warning bell for more serious ones. If it is information that needs attention, a "Warning" signal is issued. In the future, the alarm module can further communicate with the social network App like Wechat and so on, and directly send the alarm information to the Wechat of lifeguards or parents.

Through the semantic analysis of behaviors, this paper established the state table of cross-frame spatio-temporal continuous behaviors. At present, the following basic state model has been initially constructed (see Figure 16). If the swimmer has been in the "ok" state, it remains unchanged. If entering the "underwater", the timing will start. If it has timeout (such as 8 seconds, the time can be configured), the alarm "SOS" will be issued. If it is a long run, a warning message "warning" is issued. For simplicity, if a state switch occurs, it is retimed (this part of the state transition diagram is not shown). At present, it is mainly to verify the design scheme of this project, and can be further expanded according to the behavior in the future.

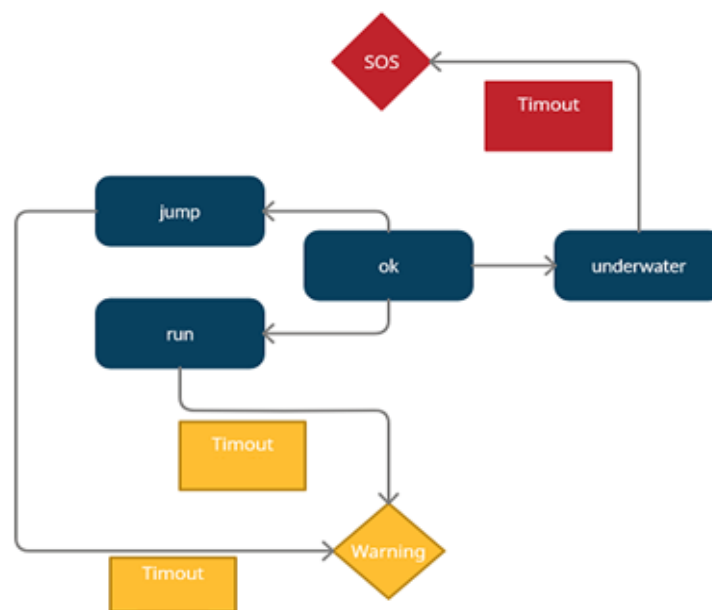


Figure 16. Finite State Model

Now, we propose Cross-Frame Modeling algorithm (Figure 17) to track the swimmer with spatio-temporal information.

Algorithm: Cross-Frame Modeling

Initialization:

Initialize the *statemodel*, which is key-value data structure, $\langle id, (state, statedelay) \rangle$. *statemodel* is used to track every swimmer ID's state and hold time in that state.

Cross-Frame Modeling: For the swimmer ID in every frame of the input video, we record this ID's *state* and hold time *statedelay* in this state. Note: if state is changed, *statedelay* is reset to 0.

```
1: while cmd != "quit" do
2:   for every swimmer ID in the current frame do
3:     switch (statemodel[id].state):
4:       case "ok" : // the current state is "ok"
5:         nothing to do, continue to track
6:       case "underwater" : // the current state is "underwater"
7:         if statemodel[id].statedelay > SOSTIMEOUT then //such as 8 seconds
8:           send the SOS warning alert!
9:         endif
10:      case "jump" : // the current state is "jump"
11:        if statemodel[id].statedelay > JTIMEOUT then //such as 3 seconds
12:          send the jump warning alert!
13:        endif
14:      case "run" : // the current state is "run"
15:        if statemodel[id].statedelay > RTIMEOUT then //such as 5 seconds
16:          send the running warning alert!
17:        endif
18:    end switch
19:  end for
20: end while
```

Figure 17. Cross-Frame Modeling Algorithm

Figure 18 below tracks two swimmers distinguished by their IDs, even as their states change while their IDs remain the same. For example, Swimmer 1 is always "ok", but Swimmer 2 is sometimes over the water and sometimes underwater. If it is underwater, the timing starts. And once time out, a warning is issued according to the state model.



Figure 18. Dynamic tracking of the swimmer's state

2.6 Swimmer Following

First, we utilize the vehicle's onboard Lidar to build the map of the swimming pool (Figure 19b), obtaining information about the pool's shape and any obstacles along its edges. Next, we manually mark the boundaries of the swimming pool to prevent the vehicle from leaving them.

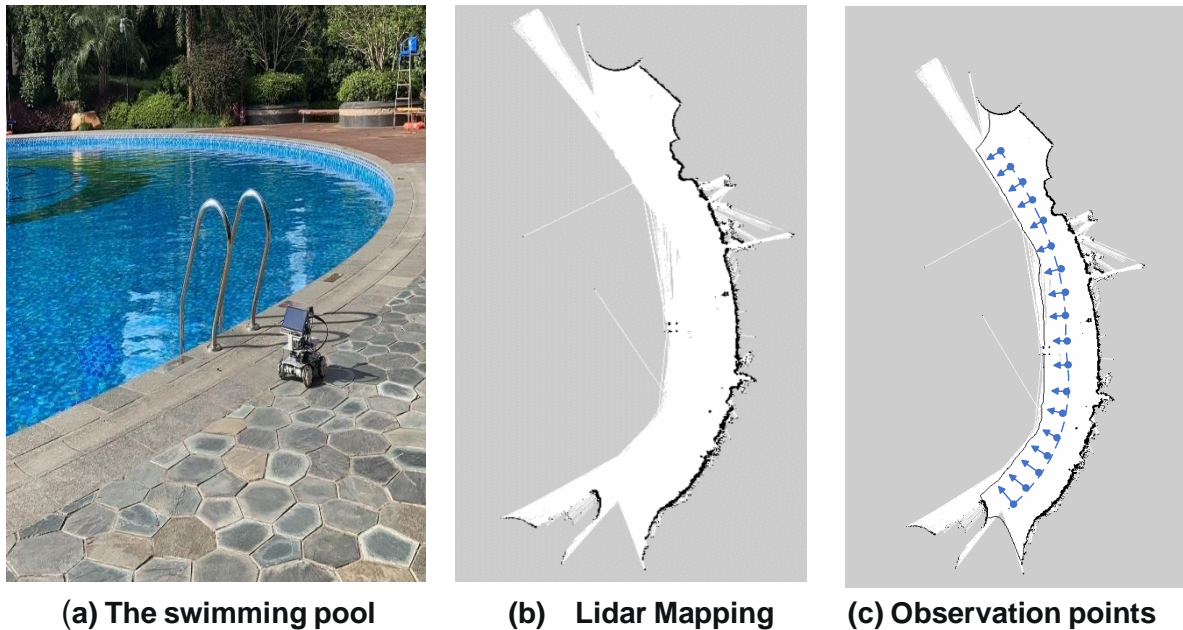


Figure 19. Lidar Mapping and Observation Points

Afterward, we designate a series of observation points of the planned path on this map (Figure 19c), including their coordinates and observation angles. These observation points serve as navigation targets for the vehicle in the subsequent following algorithm.

Regarding the Swimmer ID to be followed, in general, the vehicle is in one of three states: forward, stop, or backward (Figure 20). If the swimmer is shifted to the right and forward, we should let the vehicle to go to the next observation point (case 1 in Figure 20, also see Figure 21a).

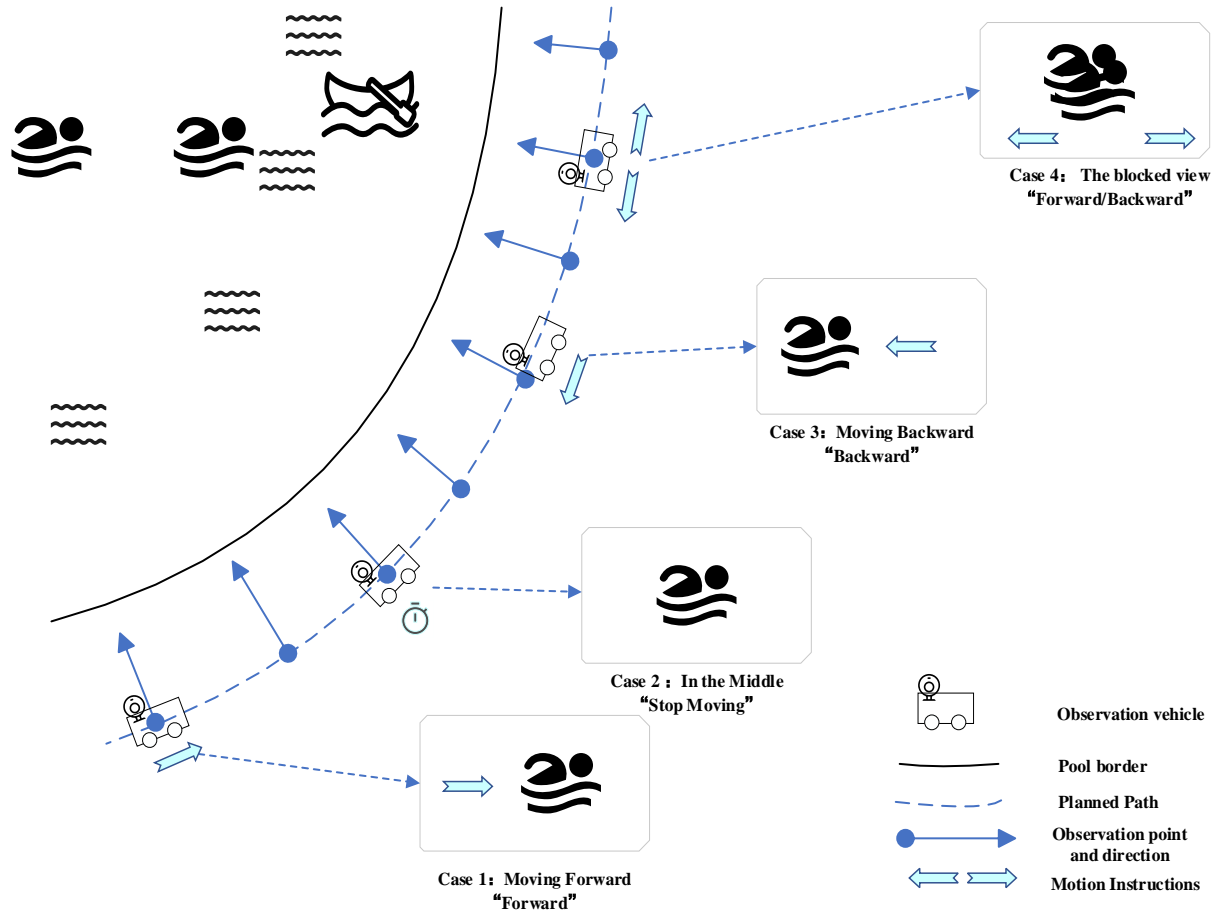


Figure 20. Swimmer Following

If the current observation position is exactly in the middle then we stop moving (case 2 in Figure 20, also see Figure 21b). If the tracked ID is disappeared (blocked by some objects, case 4 in Figure 20, also see Figure 21c), we wait for a while. If a timeout occurs, we adjust the vehicle's direction by moving forward or backward based on the previous motion direction, attempting to re-follow the target (Figure 21d). If re-follow is unsuccessful, a warning message is issued.

For our ROS2 (Robot Operating System) ⁹ based vehicle, the *move_base_simple/goal* is a ROS topic used for publishing simple messages of movement goals. It is of type *geometry_msgs/PoseStamped*, which contains pose information for the target location, including three-dimensional coordinates and quaternions. Every observation point includes the pose information for the target location. So we just simply publish this pose information to *move_base_simple/goal* to navigate the vehicle to the next target.

⁹ <https://docs.ros.org/>



(a) Moving Forward



(b) In the Middle "Stop Moving"



(c) The blocked view



(d) Re-follow the target

Figure 21. Different Swimmer Following Cases

The vehicle is equipped with the ROS2 runtime environment, which includes built-in navigation and obstacle avoidance algorithms. Based on the discussion above, Swimmer Following algorithm is proposed (Figure 22). When we provide the coordinates and angles of the target observation points, the vehicle's onboard navigation algorithm will move the vehicle to the desired location. If any obstacles are detected by Lidar, such as a person passing by, the navigation algorithm will automatically perform obstacle avoidance, simplifying our algorithm's design. Additionally, we intermittently move based on these observation points, meaning the vehicle is not constantly in motion. This reduces energy consumption and increases the vehicle's observation time.

Algorithm: Swimmer Following

Initialization:

Load the Planned Path array *poselist*.

According to the current position to determine the position index *poseindex* of the closest observation point (usually the *poseindex* is 0).

Swimmer Following: If the swimmer ID is specified to track, then for this ID in every frame of the input video, we record this ID's movement *center_position*, movement *direction*, and hold time in that direction of movement *Duration*.

```
1:  while cmd != "quit" do
2:      switch (center_position):
3:          case "forward" : // the center_position is shifted to the right and forward
4:              poseindex = poseindex + 1 //move forward to the next observation point
5:              motioncontrol("forward", poseindex)
6:          case "center" : // the center_position is in the middle of the frame
7:              motioncontrol("stop") // The current observation position is exactly in
8:                  // the middle and we stop moving
9:          case "backward" : // the center_position is shifted to the left and backward
10:             poseindex = poseindex - 1 //move backward to the observation point
11:             motioncontrol("backward", poseindex)
12:          case "blocked" : // the tracked ID is disappeared (blocked by some objects)
13:             if Duration > WaitTime then // we wait for the seconds (WaitTime)
14:                 if direction == "forward" then
15:                     poseindex = poseindex + 1 //continue to move forward
16:                     motioncontrol("forward", poseindex)
17:                 else
18:                     poseindex = poseindex - 1 //continue to move backward
19:                     motioncontrol("backward", poseindex)
20:                 endif
21:             endif
22:          end switch
23:      end while
```

Figure 22. Swimmer Following Algorithm

2.7 Mosaic-based privacy protection

For the output video (Figure 23a), using OpenCV, we apply Mosaic processing to each swimmer ID (Figure 23b). This prevents the leakage of swimmers' privacy in public pools. Depending on specific requirements, we can also adjust the granularity of the Mosaic processing. A coarser granularity results in more blurred images of swimmers. Furthermore, to meet the needs of personalized care form

parents, we can designate the swimmers to be tracked for individual users, and their images will not undergo Mosaic processing separately (Figure 23c). This allows parents to promptly assess the state of the specified swimmer.

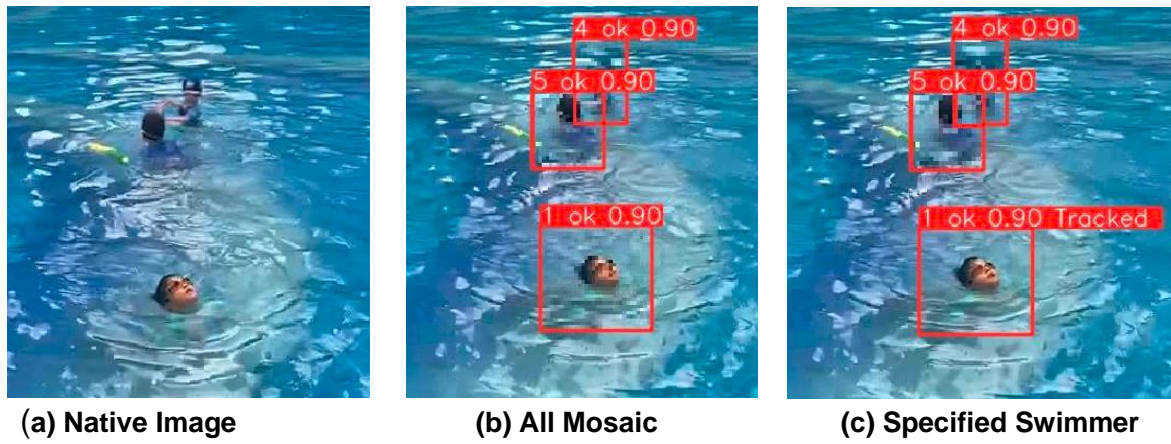


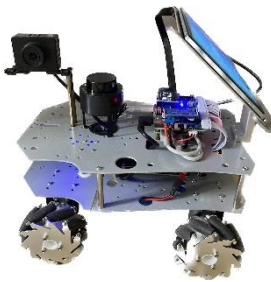
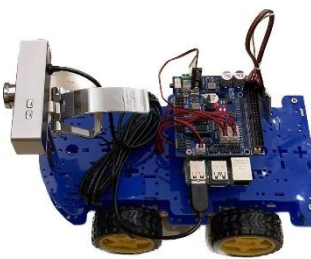
Figure 23. Mosaic-based Privacy Protection

3. Evaluation

3.1 Edge Server & Vehicle Configuration

To evaluate the system's performance under various computing configurations, we set up the experimental devices as detailed in Table 3. V2 does not have a laser radar and cannot navigate. This device is primarily used to evaluate the inference performance of embedded platforms.

Table 3. Edge Server & Vehicle Configuration

Edge Server	Configuration	Vehicle	Configuration
S1: x86 CPU+GPU	Intel i7-11700@2.5GHz, 32GB RAM, NVIDIA RTX A2000	V1: WHEELTEC R500 	Jetson nano 4GB, ARM cortex-A57@1.43GHz, 4GB RAM, N10P Lidar
S2: x86 CPU only	Intel i7-8565U@1.8GHz, 32GB RAM	V2: RaspBerry Pi 4B 	RaspBerry Pi 4 model B, ARM Cortex-A72@1.5GHz, 8GB RAM
S3: Apple M2	ARMv8-A64@3.5GHz, 16GB RAM		

3.2 Lightweight YOLO Model

Since the DeepSORT-based algorithm does not involve the accuracy of the swimmer experimental data for the traditional algorithm (except for the ReID module, which uses a simple CNN inference network) and cross-frame modeling, its speed and calculation amount are also relatively fixed, and the effect of its performance depends on the accuracy and calculation amount of the lightweight target detection algorithm. So the YOLO model plays an important role in the whole framework. Therefore, three object detection models are mainly evaluated. The first Baseline model was trained by the predefined model of YOLOv5s. The second is a lightweight network (Ghostnet+) with the addition of EIOU loss function, ACON-

C activation function, and Ghostnet; The third is Ghostnet+ plus the aforementioned BiFPN feature fusion plus coordinate attention mechanism CA fusion model (BiFPN+CA), and the fourth and fifth are tiny and small models of recent YOLOv8, respectively. Here is the basic information for the five models (the first three were trained for 400 epochs each, and the last two were trained for 500 epochs each according to the documentation.

Table 4. Comparison of Different Network Models

Model Type	Size (MB)	Layers	Parameters	FLOPS	P	R	mAP @.5
Baseline (YOLOv5s)	14.04	384	7,429,569	16.2	0.74	0.51	0.51
Ghostnet+	8.34	583	4,003,273	8.4	0.71	0.46	0.48
BiFPN+CA	12.76	448	6,291,121	14.0	0.74	0.49	0.48
Small (YOLOv8)	22.5	168	11,127,132	28.4	0.73	0.491	0.489
Tiny (YOLOv8)	6.2	168	3,006,428	8.1	0.437	0.644	0.485

On the same training platform, in addition to the model parameters and network, it can be seen that the BiFPN+CA network is lighter than Baseline, but the model accuracy is about the same as Ghostnet+. The Small model of YOLOv8 is the largest and has a large amount of calculation, but the accuracy and recall rate are not improved. The Tiny model has the smallest model and the smallest amount of calculation, but the lowest accuracy and large loss. In general, while Ghostnet+ is the most lightweight and has little loss of accuracy, we mainly choose Ghostnet+ as our object detection network. We also focus on inference speed. The following table shows a comparison of Ghostnet+, Baseline, and YOLOv8 Small/Tiny models in terms of overall speed, including inference speed for object detection and processing speed for state tracking based on DeepSORT:

Table 5. Comparison of Inference Speed

Model Types	Object Detection (ms/FPS)	State Tracking (ms/FPS)
Baseline (YOLOv5s)	8.6/116.3	88.4/11.3
Ghostnet+	25/40	45.1/22.2
Small (YOLOv8)	9.6/104.2	21.2/47.2
Tiny (YOLOv8)	6.9/144.9	19.7/50.7

It can be seen from the above table that at present the whole processing speed basically reaches more than 20FPS, which can meet the requirements for monitoring, while Ghostnet+ is slower in reasoning, but the overall processing speed is not slow. YOLOv8 has the advantage in speed, but Small model is large (the model size is 22.5MB), and YOLOv8 Tiny model is not accurate enough. So, we select Ghostnet+ as the object detection network at the present time.

3.3 Local vs. Distributed Deployment

According to the configuration of Table 3, the performance of both local and distributed deployment (Figure 6) are tested and summarized in Table 6. For local deployment, V1 (Jetson nano) has a relatively high computing power. Despite being an embedded platform, it has an NVIDIA Maxwell GPU. So the state detection and Cross-Frame Modeling require an average of 0.14 and 0.13 seconds, respectively. This results in a total speed of approximately 4 FPS, which only barely meets the requirements for swimmer tracking. However, for V2 (RaspBerry Pi), it relies primarily on CPU computation, resulting in a significantly lower speed of around 0.2 FPS. While reducing the image size to 320 (default is 640 for the camera) can increase the FPS to approximately 0.5, but this size has shown a decrease in accuracy.

For distributed deployment, if the Edge Server is S1 and we use the lightweight Ghostnet+ network, the total FPS approaches 30 FPS, resulting in a more effective swimmer tracking. From this, it is evident that distributed deployment offers superior performance and is suitable for scenarios where vehicle computing power is constrained. We observed that in certain scenarios, Ghostnet+ performed less effectively than the Baseline. We speculate that this may be due to suboptimal operators in the lightweight network, which could be explored as a potential direction for further research.

Table 6. Local vs. Distributed Deployment (Default imgsize = 640)

Deployment Types		Detection (s)	Modeling (s)	FPS	
Local	V1 (Jetson nano)	Baseline	0.14	0.13	3.7
		Ghostnet+	0.19	0.13	3.1
	V2 (RaspBerry Pi)	Baseline	5.1	0.55	0.18
		Ghostnet+	5.6	0.55	0.16
		Baseline (imgsize = 320)	1.35	0.55	0.53
		Ghostnet+ (imgsize = 320)	1.55	0.56	0.47
Distributed	S1+V1 (RTX A2000)	Baseline	0.07	0.014	11.9
		Ghostnet+	0.02	0.014	29.4
	S2+V1 (CPU only)	Baseline	0.69	0.17	1.16
		Ghostnet+	0.6	0.17	1.3
	S3+V1 (Apple M2)	Baseline	0.143	0.014	6.34
		Ghostnet+	0.16	0.014	5.75

To further investigate the performance of the Raspberry Pi, we conducted separate tests with two target detection models: YOLOv5-lite, designed for

embedded platforms (<https://github.com/ppogg/YOLOv5-Lite>), and ByteTrack, a fast end-to-end multi-object tracking model (<https://github.com/ifzhang/ByteTrack>).

On the Raspberry Pi, under the same resolution, the performance of PyTorch-based models is not very satisfactory. Only the state detection, which requires approximately 2 seconds, consumes about 2.9 GFLOPs. Similarly, running the end-to-end ByteTrack model yields slightly better results, with target detection and tracking combined taking around 2 seconds, which is an improvement compared to YOLOv5-Lite.

From this observation, without optimization for embedded platforms, the detection speed remains relatively low and fails to meet the requirements. Therefore, we are considering the NCNN optimization library¹⁰, which is specifically designed for embedded platforms and utilizes C++ programming, providing faster performance compared to Python-based PyTorch.

We conducted preliminary tests, and the built-in YOLO algorithm provided by the NCNN system exhibited excellent performance, as shown in Table 7 below. With optimization, we anticipate target detection to be under 100 milliseconds.

Table 7. The performance of the NCNN library on the Raspberry Pi

Model Types	Min (ms)	Max(ms)	Average(ms)
mobilenet_yolo	785.36	811.09	800.64
yolov4-tiny	537.07	553.48	541.82
nanodet_m	151.50	154.25	152.44
yolo-fastest-1.1	94.88	101.02	95.94
yolo-fastestv2	62.08	63.34	62.64
vision_transformer	7140.38	7293.43	7206.64
FastestDet	70.20	71.90	70.86

From Table 7, it is evident that there are significant computational limitations on embedded platforms currently, and the computing requirements for Transformer-based object detection are even higher. Even with hardware optimizations like NCNN, it can only run very lightweight networks such as yolo-fastest, often at the cost of reduced accuracy. Therefore, the current distributed deployment approach proves to be a more viable solution for swimmer tracking.

3.4 Cross-Frame Modeling

The actual effect analysis is carried out below. In a clear background, the status of the main task can be tracked and recorded, but the fourth ID in the figure is not recognized due to the small picture. The ID of the swimmer is also lost in tracking (but due to dynamic tracking, the temporarily lost ID will be found in the follow-up. For example, the swimmer ID lost in the second panel is retrieved in the

¹⁰ <https://github.com/Tencent/ncnn>

third panel and remains unchanged.) It can be seen that the state dynamic tracking of ID is basically realized, but there are still some flaws due to the accuracy of ReID and target detection, and the data set and model tracking algorithm need to be further improved.

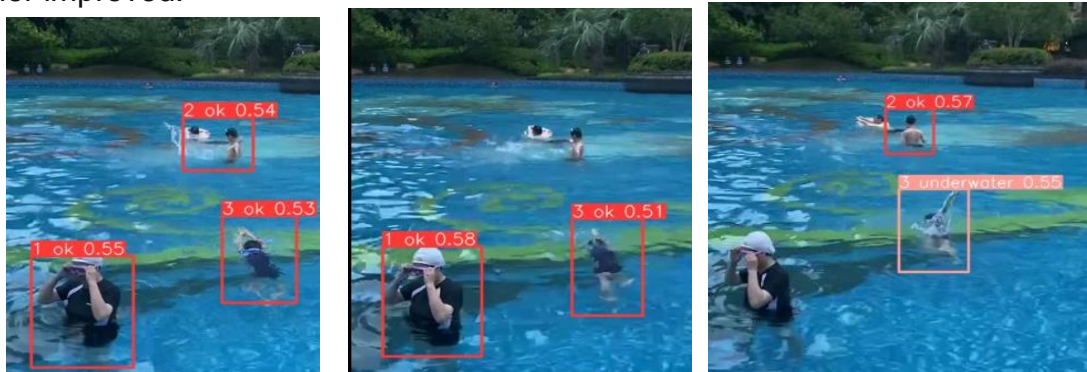
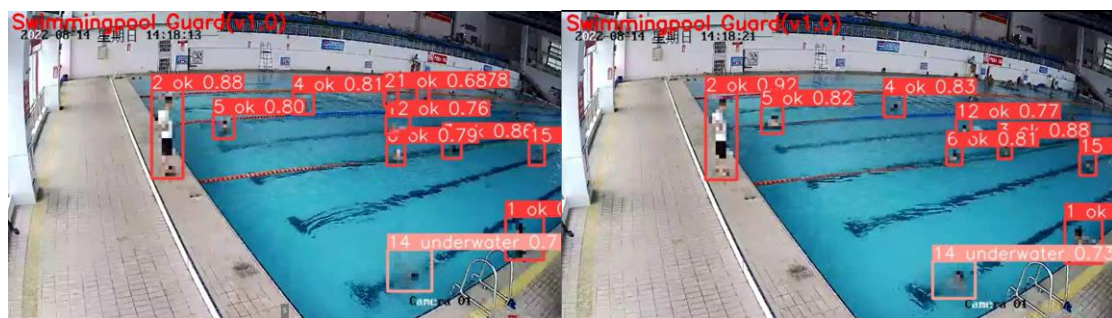


Figure 24. Swimming Pool Behavior Tracking

I downloaded drowning videos from the Internet and used our system for analysis. When a swimmer is "underwater" for more than 10 seconds (this parameter can be modified), the system marks the swimmer as "SOS" and a warning bell is sounded to alert everyone.

We conducted tests on drowning videos from the Internet to verify its dangerous behavior detection effect (Figure 1). The state of the swimmer 14 has gone through the process of "underwater" (at 14:18:13, Figure 25a) -> "underwater" (SOS timeout alarm, at 14:18:21, Figure 25b), which can track the state of swimmers. According to the state transition modeling, once the underwater timeout (tentatively 8 seconds here), the alarm can be reported in time, and the design goal has been initially achieved.



(a) Underwater

(b) Underwater time-out alarm

Figure 25. Swimming pool dangerous behavior detection effect

In order to compare the effect of the Object Tracking algorithm (Figure 26), we also compared with the current end-to-end object tracking algorithm (ByteTrack¹¹).

¹¹ <https://paperswithcode.com/sota/multi-object-tracking-on-mot17>

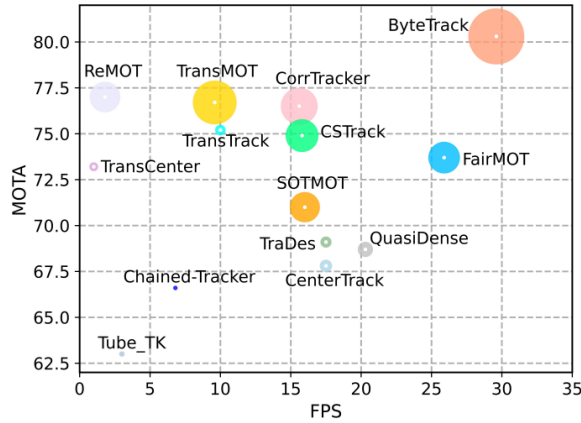
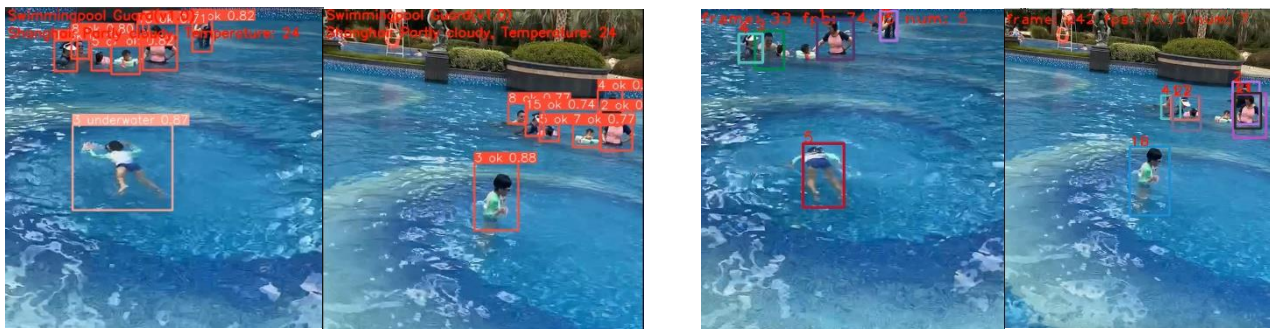


Figure 26. Accuracy and speed comparison of the ByteTrack algorithm

After practical testing, we find that its detection speed is much faster than the previous framework, with 900,000 parameters, only 3.99GFLOPS of computation, over 70FPS of inference speed, and 7.4MB model size.

However, in terms of accuracy, the accuracy of small targets and occlusion, underwater and other time series is not ideal at present. For example, for the same detection video, it can be seen that the end-to-end tracking is easy to lose the target, the state of the target cannot be accurately tracked at present, and the effect recognition of small targets is not ideal. For example, as shown in Figure 27, some small targets are lost. For the tracking of the swimmer at different times, its ID is lost and a switch occurs. However, for our method ID is not lost, and it also keeps the ID when the state is switched, which can track the state of the swimmer more accurately. For end-to-end tracking, we will also further track according to the optimization of the data set and algorithm. After all, its detection speed is much higher than that of the regular model, which may be more convenient to use in lightweight occasions.



(a) Our Method

(b) ByteTrack

Figure 27. Comparison of our method with ByteTrack Tracking Method

4. Limitations & Discussions

The current solution has not yet addressed the charging of the vehicle. This can be addressed by adopting a similar approach to that of robotic vacuum cleaners, which use Lidar-based localization to periodically return to a charging station for recharging.

Furthermore, the current distributed deployment primarily offloads core algorithms to an edge server. It can be extended to have a single edge server to control multiple vehicles for more efficient pool monitoring. Additionally, the integration of vehicles and fixed cameras can be explored as a complementary approach. If fixed cameras have limited visibility, vehicles can be moved closer to the area of interest for better observation.

The current approach utilizes 2D radar mapping, but there is potential for future exploration into 3D point cloud mapping for state detection and behavior modeling based on 3D point clouds. Currently, after radar mapping, observation points are manually set, but in the future, autonomous navigation routes can be self-determined based on the map, reducing the need for manual intervention. Additionally, integration with fixed cameras can be considered, where fixed cameras can identify the pool and vehicle positions and plan observation routes.

Experimental findings suggest that the dataset plays a crucial role in target detection. For some specific scenarios, we found that the Precision of our object detection is 0.849, Recall is 0.9, and mAP@.5 is 0.909. The dataset collected for this paper is not yet extensive, and there is room for further expansion. In the future, the dataset can be expanded through methods such as pool simulation and AIGC to enhance the generalization capability of target detection.

The paper also discusses the inference performance issues on embedded devices. In addition to further improving lightweight networks, such as using neural architecture search (NAS), neural network model quantization, model compression, etc., exploring performance optimization methods based on hardware acceleration is another future direction for exploration.

Another interesting direction for improvement is enhancing human-machine interaction. Currently, alert notifications primarily rely on monitors and sounds. In the future, there is room for diversifying alert mechanisms. For instance, a swimming pool could incorporate a Swimming-ring launcher. In the event of a drowning alert, the system could launch a Swimming-ring, attracting the attention of everyone and facilitating the prompt identification of potential drowning victim.

5. Conclusion

This paper introduces a novel swimmer following vehicle designed to dynamically track swimmers' behavior. The integration of swimmer tracking via deep learning and risk alerts using cross-frame modeling establishes an early warning system. Compared with existing deep learning methods based on object detection from

fixed cameras, we add dynamic behavior modeling of cross-frame spatio-temporal information semantics on a mobile vehicle, to address the issues like target occlusion and the tracking of distant small targets. This vehicle can be used by parents for remote monitoring of their children, or it can be temporarily rented at public swimming pools to assist parents seeking personalized care and support pool supervisors in issuing early warning alerts. We hope this vehicle contributes to creating a safe swimming environment for children.

References

- [1] Li Xiaohong. An underwater intelligent lifesaving system for swimming pools [P]. Beijing: CN110155274B, 2021-03-16.
- [2] Liu Wenjie, Li Zhitao, Li Xinglin. Swimming pool intelligent life-saving monitoring system technology and its application in the National Swimming Center[J]. Intelligent Building Electrical Technology, 2008(01):54-57.
- [3] Huang Jiaying, Zhan Jie. Automatic swimming pool drowning alarm system based on ZigBee wireless positioning [J]. Science and Technology Innovation, 2019(13):69-72.
- [4] Qian Jun, Li Kaixia, Xu Twenty. A swimming pool anti-drowning alarm device using Internet of Things technology [P]. Jiangsu Province: CN216424705U, 2022-05-03.
- [5] Jing Mingtao, Yu Teng, Feng Mengyao, Yang Guowei. Research on Swimming Pool Drowning Detection Based on Improved Mask R-CNN[J]. Journal of Qingdao University (Engineering Technology Edition), 2021,36(01):1-21.
- [6] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.
- [7] Zheng, Zhaohui, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. "Distance-IoU loss: Faster and better learning for bounding box regression." In Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 07, pp. 12993-13000. 2020.
- [8] Zhang, Yi-Fan, Weiqiang Ren, Zhang Zhang, Zhen Jia, Liang Wang, and Tieniu Tan. "Focal and efficient IOU loss for accurate bounding box regression." Neurocomputing 506 (2022): 146-157.
- [9] Ma, Ningning, Xiangyu Zhang, Ming Liu, and Jian Sun. "Activate or not: Learning customized activation." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8032-8042. 2021.
- [10] Han, Kai, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. "Ghostnet: More features from cheap operations." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1580-1589. 2020.
- [11] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781-10790. 2020.
- [12] Hou, Qibin, Daquan Zhou, and Jiashi Feng. "Coordinate attention for efficient mobile network design." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 13713-13722. 2021.
- [13] Bewley, Alex, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. "Simple online and realtime tracking." In 2016 IEEE international conference on image processing (ICIP), pp. 3464-3468. IEEE, 2016.

- [14] Wojke, Nicolai, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric." In 2017 IEEE international conference on image processing (ICIP), pp. 3645-3649. IEEE, 2017.
- [15] Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." (1960): 35-45.
- [16] Kuhn, Harold W. "The Hungarian method for the assignment problem." Naval research logistics quarterly 2, no. 1 - 2 (1955): 83-97.
- [17] vestnikov, Roman, Dmitry Stepanov, and Aleksandr Bakhshiev. "Development of Neural Network Algorithms for Early Detection of Drowning in Swimming Pools." In 2023 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), pp. 820-824. IEEE, 2023.
- [18] Li, Keshi. "Construction method of swimming pool intelligent assisted drowning detection model based on computer feature pyramid networks." In Journal of Physics: Conference Series, vol. 2137, no. 1, p. 012065. IOP Publishing, 2021.
- [19] Jensen, Morten B., Rikke Gade, and Thomas B. Moeslund. "Swimming pool occupancy analysis using deep learning on low quality video." In Proceedings of the 1st International Workshop on Multimedia Content Analysis in Sports, pp. 67-73. 2018.
- [20] Nurtanio, Ingrid, Christoforus Yohannes, Indrabayu Indrabayu, Kelvin Kelvin, and Zulfadli Zulfadli. "Automatic swimmer counter for outdoor swimming pool." In AIP Conference Proceedings, vol. 2543, no. 1. AIP Publishing, 2022.
- [21] Lei, Fei, Hengyu Zhu, Feifei Tang, and Xinyuan Wang. "Drowning behavior detection in swimming pool based on deep learning." Signal, image and video processing (2022): 1-8.
- [22] Cepeda-Pacheco, Juan Carlos, and Mari Carmen Domingo. "Deep learning and 5G and beyond for child drowning prevention in swimming pools." Sensors 22, no. 19 (2022): 7684.
- [23] He, Xinyu, Fei Yuan, Tingzhuang Liu, and Yi Zhu. "A video system based on convolutional autoencoder for drowning detection." Neural computing and applications (2023): 1-13.

致谢

1. 论文的选题来源、研究背景

本项目选题来源于作者身边的溺水事故，由于家长及泳池安保人员疏忽，造成小区一名小孩溺水身故。当时也有泳池救生员和家长在场，但由于一时疏忽，酿成悲剧。同时根据报道，溺水是青少年最主要非正常死亡原因之一，是15岁以下儿童的仅次于脑膜炎和艾滋病毒第三大杀手，每年致死14万人。所以想是否可以通过人工智能辅助预警，减少此类事故的发生，并且能够帮助泳池管理人员能够对异常的行为进行告警。

经过调查，目前的泳池监管智能设备比较少，主要依靠救生员现场看管。当前的泳池监管方案有些采用水下摄像头（部署和维护比较困难），有些基于AI的大屏监控（目标遮挡、小目标识别困难，无法提供个性化照看等），有些采用可穿戴设备（适用场景比较单一，维护比较复杂）。泳池不仅是溺水，还有一些行为也需要识别，例如跳水、奔跑等。因此，本文提出了能够跟随游泳者的智能小车，有别于传统的基于固定摄像头的监控方法，通过小车摄像头移动采集视频，采用深度学习来识别游泳者的状态，并通过状态建模，对泳池的危险行为进行预警（例如水下时间长、奔跑等）。

同时，根据游泳者的运动方向，控制小车按照预定的观测点进行移动（事先通过激光雷达建图），这样能够近距离跟踪游泳者，避免固定位置的摄像头引起的遮挡或目标太小的问题，并能够通过网络将游泳者的实时图像传送（可以近距离观测小孩的状态，例如是否疲惫）。因此，这个小车可以帮助家长照看小孩，甚至可以远程看管小孩。为了避免公共场合的隐私泄露，将不是跟踪目标的其他游泳者打上马赛克。这个小车也可以放在公共泳池，供泳池管理人员临时出租给家长，方便不方便进入泳池的家长也可以看管小孩。

2. 每一个队员在论文撰写中承担的工作以及贡献

本项目是我在导师合作指导下完成，泳池数据收集整理和标注、算法优化、状态建模、小车组装和控制、实验和论文写作等均在导师指导下由本人完成。

3. 指导老师与学生的关系，在论文写作过程中所起的作用，及指导是否有偿

本文的第一位指导老师为上海交大杨旭波教授。杨老师在计算机视觉、机器学习和隐私保护方面提供了宝贵的指导意见，在基于OpenCV的图像处理、AIGC的数据增强、分布式部署和基于马赛克的隐私保护等方面提供了很好的专业意见，并在本项目论文选题、机器学习理论和实验等方面进行了认真的无偿辅导。第二位指导老师为复旦大学卢曦教授，卢老师是我的中学生“英才计划”的指导老师，从本文的选题、AI算法及优化、状态建模、小车跟随算法、实验和论文的写作进行了全方面的无偿指导。

4. 他人协助完成的研究成果

泳池的溺水行为调查得到了多家泳池救生员的仔细答复，在此表示感谢！特别感谢我的弟弟，暑期长期泡在游泳池里面，为本文的泳池建模和小车调试提供了实验支持。

5. 指导老师简历

杨旭波，教授，博士，博导。1998年博士毕业于浙江大学计算机辅助设计与图形学(CAD&CG)国家重点实验室。1998年至2001年于德国Fraunhofer-IMK研究所虚拟现实系做博士后。2001年至2003年在新加坡国立大学混合现实实验室工作。2003年

作为学院引进人才加入上海交通大学电信学院计算机系与软件学院，负责开展图形学、虚拟现实和数字媒体领域的科研工作，并主讲和开设了计算机图形学、游戏程序设计、数字媒体与人机交互等课程。现任中国图学学会计算图学专委会副主任委员，中国图像图形学会虚拟现实专委会委员、中国计算机学会CAD&CG专委会和人机交互专委会委员，上海市图学学会理事和图像图形学会委员。主要研究方向为计算机图形学、虚拟现实（增强现实/混合现实）、数字媒体与人机交互。在国内外学术会议和期刊上发表学术论文近百篇，作为首位中国学者担任IEEE VR 2023大会共同主席。

卢曦，复旦大学教授，美国卡耐基梅隆大学（CMU）访问学者；四川大学计算机本科、硕士和博士，复旦大学博士后。CCF 高级会员，协同计算专委会秘书长，上海市数据科学重点实验室副主任，上海市计算机学会协同与信息服务专委会副主任。研究方向为CSCW与社会计算、人机协同与人智交互、群智协同与决策、推荐系统。承担多项国家自然科学基金项目、国家科技部重点研发计划课题、863 课题和上海市项目。在 CSCW、CHI、UbiComp、NeurIPS、WWW、SIGIR、TKDE 、TOIS等顶级学术会议和期刊上发表一系列研究长文，共同获得过CSCW'15最佳论文和CSCW'18最佳论文提名奖。常规担任 CHI 和 CSCW 的 Associate Chair，担任多个重要学术会议的 PC Co-Chair，以及国内外期刊的编委和客座编辑。