Classical And Deep Learning Detection with Kalman Filter Tracking for Underwater Fish

Ganesh M[1], Kumaraguru G[2], Karpagaselvi P[3], Sakthivel Murugan S[4]
[123]*Dept. of Electronics and Communication Engineering, Sri Venkateswaraa College of Technology, India*
[4]*Dept. of Electronics and Communication Engineering, National Institute of Technical Teachers Training and Research, Chennai, India*

***Abstract*—Labelling every frame of underwater video by hand is far too slow to support behavioural studies of reef fish. This paper describes a semi-automatic workflow that starts from a handful of hand-labelled frames, trains a single-class YOLO26n detector, and then uses that detector to speed up the annotation of further frames, while a lightweight Kalman-filter tracker keeps a consistent identity for each fish, even though short occlusions behind coral. All work was done on hand-held coral-reef footage using a CPU only. Before the final design was reached, three classical detectors (Gaussian mixture background subtraction, colour and contrast blob detection, and a gradient-boosting classifier on HOG features) and two trackers (ByteTrack and BoT-SORT) were tested, and the reasons each fell short are reported as negative results. The final detector was trained on 879 images, including fish-free frames used as background examples, and reached a precision of 0.671, a recall of 0.667 and an mAP50 of 0.679 on held-out validation images. Adding the Kalman tracker gave a tracking accuracy of about 90 percent on the first video, and a test on a second video showed that the method carries over reasonably well. Growing the annotated dataset improved results far more than training for longer. Occlusion, fast motion and detection quality remain the main sources of error.**

***Index Terms*—underwater video, fish detection, multi-object tracking, YOLO, Kalman filter, annotation, background subtraction, ByteTrack, BoT-SORT**

I. INTRODUCTION

Video is one of the most practical ways to watch fish without disturbing them, and it supports population counts, behavioural studies and reef-health assessment. The difficulty lies in the analysis. Marking every fish in every frame is slow, and the workload grows quickly with video length; large coral-reef video collections such as Fish4Knowledge [1] show clearly why this cannot be done by hand at scale.

Underwater footage is also harder to process than ordinary surveillance video. The camera is usually carried by a diver and is never still, fish and coral have similar colours and low contrast, light is attenuated and scattered by the water, and the fish themselves are small and often swim in dense, overlapping groups. Each of these problems affected the choices described in this paper.

We built and tested a complete workflow that runs from raw video to a trajectory for every tracked fish. It was designed around three goals: (1) to reduce the manual annotation effort by training a detector on a small seed set and letting it propose labels that a person only needs to check and fix; (2) to obtain a reliable single-class ("fish") detector that copes with changing light, different species colouring and camera motion; and (3) to keep a stable identity for each fish, including after it disappears briefly behind coral. Everything was developed on a laptop-class CPU (12th Gen Intel Core i5) without a GPU. That limit shaped many decisions, such as timing one training epoch before starting a full run and choosing a motion-only tracker over a heavier appearance-based one.

The main contributions are: (i) a staged comparison of classical detectors on hand-held reef footage, with the reasons for their failures documented; (ii) a semi-automatic annotation and training procedure for a YOLO26n detector, including the use of fish-free background frames; (iii) a comparison of ByteTrack, BoT-SORT and a custom Kalman-filter tracker, together with the tuning problems met along the way; and (iv) a robustness check on a second underwater video. Section II reviews the approaches considered, Section III gives the system overview, Section IV

describes the methods, Section V reports the analysis and results, and Section VI concludes.

II. APPROACHES CONSIDERED

Moving-object detection with an adaptive Gaussian mixture model (GMM) [2] assumes a fixed camera. Hand-crafted descriptors such as histograms of oriented gradients (HOG) [3] combined with boosted classifiers [4] can find objects in a single frame without motion cues, and deep one-stage detectors of the YOLO family [5] learn the features directly from labelled images.

For tracking, SORT [6] pairs a Kalman filter [7] with frame-to-frame association. ByteTrack [8] extends this idea by also using low-confidence detection boxes during association, and BoT-SORT [9] pairs motion prediction with a learned appearance model for re-identification (re-ID). The following methods were implemented and compared in this work: a fine-tuned YOLO26n detector as the main detector; MOG2 background subtraction and two further classical detectors as training-free baselines; ByteTrack and BoT-SORT as ready-made trackers; and a SORT-style Kalman tracker written for this project, which was adopted as the final tracking method.

III. PROPOSED SYSTEM

The system follows a modular detection-and-tracking design in which fish localisation and multi-object tracking are separate stages that can be evaluated independently (Fig. 1). Each frame is first pre-processed and passed to the detector, which proposes fish boxes. Duplicate proposals on the same fish are removed, and the remaining boxes go to the Kalman-filter tracker, which assigns persistent identities.

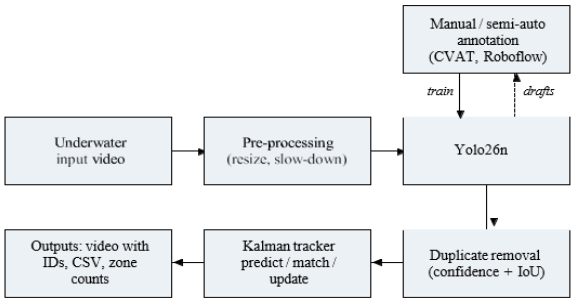


Fig. 1. Overall detection and tracking pipeline, with the annotation loop that feeds the detector.

The outputs are an annotated video showing the fish IDs, and a CSV file holding the frame-level detection and tracking records. Annotation runs alongside this loop: the detector drafts boxes on new frames, a person corrects them, and the corrected frames are used to retrain the detector.

IV. METHODOLOGY

A. Video Data

The first study video has 911 frames at 1280×720 and 30 fps. It was filmed by hand over a coral reef and shows stationary coral, open water and small reef fish in yellow, blue and grey or dark colours that move independently of the camera (Fig. 2). A second, higher-resolution video (3840×2160) was obtained later, and a further underwater video was used for the robustness test in Section V-D.



Fig. 2. Sample frame from the coral-reef video.

B. Classical Detection Baselines

1)GMM background subtraction:

MOG2 [2] from OpenCV [10] was applied per pixel, followed by morphological opening and closing, and the resulting contours were filtered by size and shape (area and aspect ratio). Because the camera was hand-held, the whole frame registered as foreground and the stationary coral produced a large number of false positives. We then compensated for camera motion by estimating frame-to-frame homographies from ORB feature matches [11] and warping every frame into a common reference. This exposed two structural limits. First, the overlap with a single fixed reference shrank as the camera drifted, and detection dropped to almost nothing after about three seconds; resetting the reference whenever the overlap fell below 60% of the frame area restored detection, at the price of a short relearning period after each reset. Second, GMM responds to pixel change and not to object identity, so swaying coral and moving caustics were detected as

readily as fish, while a fish that paused was slowly absorbed into the background model and lost. Tuning the learning rate and a blob-solidity filter reduced both problems but did not remove them.

2)Colour and local-contrast blobs:

Next, a detector that looks at each frame on its own was tried, so that camera motion no longer mattered. A colour threshold picked out saturated yellow and blue bodies, which are uncommon on coral, and a black-hat (median-background difference) filter picked out small dark blobs in smooth open water, with a Sobel texture mask used to suppress responses on textured coral. Connected-component statistics then removed regions that did not look like fish. Precision was better than with GMM, but small or camouflaged fish were missed and coral edges still produced false detections.

3)HOG with gradient boosting:

Ground truth came from a reference video with manually checked boxes and persistent fish IDs, exported from an outside annotation and tracking tool. The box coordinates were recovered programmatically from the overlay graphics of that video, giving 1,379 verified boxes. A 32×32 HOG descriptor [3] was computed for each box, and negative samples came from regions proposed by a multi-scale difference-of-Gaussians detector wherever these missed every ground-truth box. We trained a histogram-based gradient-boosting classifier [4], [12] on 29,421 class-balanced samples. Tested on later frames (600 onward) that were held out in time, it reached an ROC-AUC of 0.92 and 91% precision at a probability threshold of 0.5. Non-maximum suppression with an IoU of 0.3 was then applied to the per-frame output. Hand-crafted features still limited what the classifier could learn on dense schools, which motivated a deep detector.

C. Annotation and Training Data

Training images came from five sources (Table I). Five frames were annotated by hand first, and species labels were merged into one “fish” class. A 5-species set exported from Roboflow [13] added 125 images. A collaborator then annotated 612 frames in CVAT [14], of which 429 had matching image and label files; frames without any visible fish were kept as background examples. A second manual round in Roboflow contributed 320 more frames. A 23-species open dataset of 4,120 images was also tried and then excluded.

TABLE I TRAINING DATA SOURCES

Source	Images	Handling
Manual, first pass	5	Species labels merged to one class
Roboflow export (5 species)	125	Merged to one class
CVAT annotation	429	Single class; fish-free frames kept as negatives
Second manual round (Roboflow)	320	Single class
Open 23-species set	(4,120)	Excluded: different camera and environment

A merge script combined the selected sources, remapped classes where needed, and split the data 85/15 into training and validation parts using a fixed seed. Importantly, fish-free frames were retained as explicit negative examples. The earlier detector tended to label swaying coral and background texture as fish, and showing the model frames with no fish is a standard way to reduce such false positives; it did help here. The final set had 879 images (5 + 125 + 429 + 320), split into 748 for training and 131 for validation, where 131 of the training images and 22 of the validation images contained no fish.

The 23-species dataset was left out because it came from a visibly different camera and environment, and we judged that it would dilute rather than help in-domain accuracy. We did not test this by ablation; the decision followed from our shift towards annotating the study video itself. Two other sources were also rejected. A collaborator had run an RF-DETR Nano detector and shared only its JSON output and a video with the boxes drawn into the pixels. The JSON pointed to a source video that could no longer be found, and the burned-in boxes would have become part of what a model learned to detect. The practical lesson is to always keep the clean source video together with any machine-generated labels.

D. Detector and Training

The detector is YOLO26n (nano) from the Ultralytics framework [15], initialised from weights pretrained on COCO [16] with the detection head re-initialised for the single class “fish”. The five lowest backbone

layers were held fixed while fine-tuning (freeze=5). This keeps general low-level features, as discussed in [17], while the later layers adapt to underwater images, whose colour balance and contrast differ strongly from natural images.

Training used 640×640 inputs and the optimiser chosen automatically by the framework, on a 12th Gen Intel Core i5 CPU. Because there was no GPU, one epoch was timed before every full run to set an affordable epoch budget; the time per epoch grew from about 1.1 minutes for 130 images to about 5 minutes for 879.

E. Multi-Object Tracking

1) *ByteTrack and BoT-SORT:*

ByteTrack [8] was used first as the baseline tracker. BoT-SORT [9] was also run with its default settings and showed a clear weakness: a fish hidden briefly behind coral came back with a new ID. Raising the track_buffer value from 30 to 150 frames and switching on appearance re-ID (with_reid=True) addressed this, but at the cost of an extra embedding model, which is a heavy burden on a CPU-only machine.

2) *Kalman-filter tracker:*

We therefore wrote a SORT-style tracker [6] with no appearance model. Each fish has its own constant-velocity Kalman filter [7] over the box centre, with state vector $[x, y, v_x, v_y]$ and a two-dimensional position measurement:

$$x_k = F x_{k-1} + w_k \quad (1)$$

$$z_k = H x_k + v_k \quad (2)$$

Here F moves the position forward by the velocity over one frame, H selects the position from the state, and w_k and v_k are zero-mean process and measurement noise with covariances Q and R. On every frame each active track predicts its next position; detections are matched greedily to predictions by distance, inside a maximum gating distance; matched tracks are corrected with the detection; unmatched tracks continue on prediction alone for a fixed number of frames before deletion; and unmatched detections start new tracks. A track is only displayed once enough detections have confirmed it, which filters out one-frame noise.

Two implementation problems were found and fixed. First, the first choice of Q and R made the filter trust its own prediction more than the new detection. During fast motion the track lagged behind the fish, matching failed, and duplicate IDs appeared. Giving more weight to the measurements cured this, and the fix was checked on synthetic motion and occlusion cases before use.

Second, two boxes sometimes covered one fish because they just failed to merge in the detector's own non-maximum suppression, which created a second ID on the same animal. A deduplication pass was added before association: detections are sorted by confidence, and any box whose overlap with an already kept box is high enough is discarded. Overlap is measured by

$$\text{IoU} = \text{overlap area} / (\text{area}_A + \text{area}_B - \text{overlap area}) \quad (3)$$

F. Pre-processing of High-Resolution Video

Two measures were added for the later 3840×2160 video. The frames were reduced to 1280×720 before detection, because the detector resizes its input to 640×640 anyway, so full 4K gave no accuracy gain and only cost time. The video was also slowed to half speed by writing each frame twice, which shortens the distance a fish moves between processed frames and makes distance-based matching easier and ID switches less frequent.

G. Zone-Based Spatial Aggregation

In crowded scenes where individual fish cannot all be separated; density is still useful.

The frame is divided into a 3×3 grid of zones, each detection is assigned to a zone by its centre, and the count per zone and per frame is written to a CSV file and drawn on the video. High-density regions can therefore be found without identifying every fish in them.

V. ANALYSIS AND RESULTS

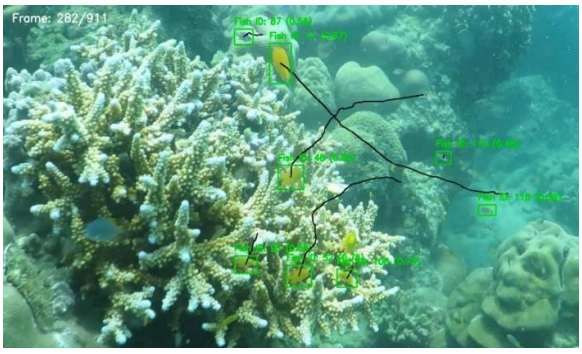
The tracking work was organised as four successive studies, each adding a tool to deal with the main limit of the previous one (Table II). The classical detectors of Section IV-B came first; the studies below use the detectors and trackers of Sections IV-D and IV-E.

TABLE II SUMMARY OF THE TRACKING STUDIES

Study / tool	Observed output	Remaining limitation
Baseline: ByteTrack	IDs and tracks produced, but IDs inconsistent	ID errors on overlap, fast motion, missed detections
Detector: Roboflow annotation and YOLO26n training	More reliable fish detections	Tracks lost on occlusion, rapid motion, missed frames
Motion: Kalman filter	Smoother, more continuous tracks; accuracy about 90%	Errors on occlusion, sudden movement, detection loss
Robustness: combined method, new video	Short losses bridged; trajectories mostly continuous	Depends on motion, occlusion, background, detection quality

A. Baseline Tracking: ByteTrack

ByteTrack produced IDs and trajectories for the whole video.



ByteTrack output showing fish IDs, confidence scores and trajectories.

Errors appeared when fish overlapped, changed position quickly or were missed for a frame or two by the detector, and these led to wrong or changing IDs. Tracking quality was bounded by detection quality, so attention turned to the training data.

B. Detector Improvement: Annotation and YOLO26n

Roboflow and CVAT were used to annotate, review and correct fish boxes before training. Their role was to give the detector better training data, not to replace the tracking algorithm. Detections became more reliable, but in video a fish that was hidden, moving fast or missed in one frame could still lose its track.



Detections from the trained model

Table III lists the training runs. The first run was validated on its own five training images, so its high mAP50 of 0.526 only shows overfitting and is not an accuracy estimate. The final run, on 748 training and 131 validation images, is the first one that includes background-only frames, and it is the best confirmed result: precision 0.671, recall 0.667 and mAP50 0.679 on held-out data. Metrics of the two intermediate runs were not recorded and are not estimated here.

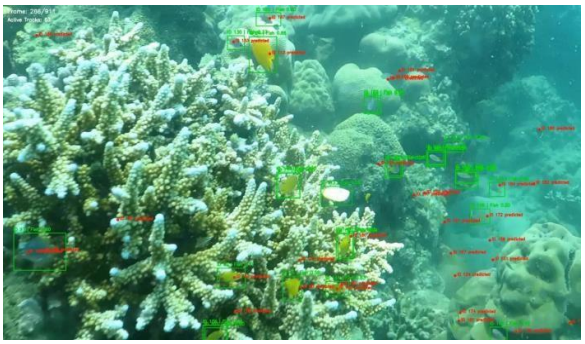
TABLE III TRAINING RUNS OF THE DETECTOR

Run	Train / val	Epoch s	P	R	mAP50	mAP50–95
Initial	5 / 5 (same)	60	0.044	0.941	0.526	0.360
Intermediate 1	110 / 19	80	n/r	n/r	n/r	n/r
Intermediate 2	476 / 83	30	n/r	n/r	n/r	n/r
Final	748 / 131	40	0.671	0.667	0.679	0.284

In Table III, n/r means not recorded. The loss curves of the final run were still falling at epoch 40, and mAP50–95 rose from 0.263 to 0.284 between epochs 30 and 40, so the model had not fully converged. Even so, the large gain came from more and better data: precision rose from 0.044 to 0.671 as the training set grew from 5 to 879 images, whereas extra epochs on a fixed set gave only small gains. Detection alone ran at about 60 ms per frame (16–17 FPS) on the CPU, not counting tracking.

C. Motion Prediction: Kalman Filter

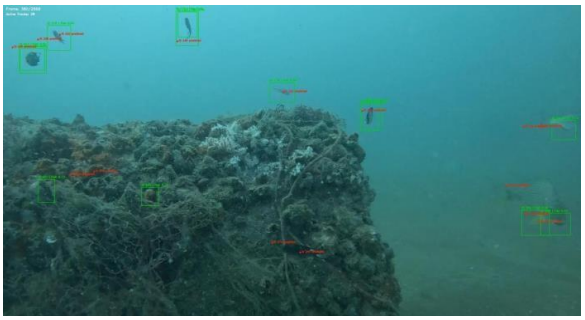
With the Kalman filter, tracking improved to an accuracy of about 90% on the first video. The predicted positions, shown in red, covered frames in which the detector found nothing, so tracks stayed smooth and continuous. Some errors remained during occlusion, sudden fish movement and longer detection loss, so the Kalman tracker was combined with the trained detector and tested on a new video.



Kalman-filter tracking green boxes are detections and red markers are predicted positions.

D. Robustness on a Second Video

The combined method was run on a second underwater video to see whether it still produced reliable trajectories on different footag. The filter bridged short detection losses and kept trajectories mostly continuous, although fish motion, occlusion, background and detection quality all still affected the result. Performance was therefore consistent with the first video but not perfect, which points to a detector trained on more and denser annotation.



Combined method (trained detector and Kalman tracker) on the second video.

E. Discussion

Three points stand out. First, the weak step in every earlier pipeline was detection, not association: ByteTrack, BoT-SORT and the Kalman tracker all improved when the detector improved. Second, Kalman tracking quality depends strongly on the balance between Q and R. A wrongly weighted filter produced identity failures that looked like detector or matching errors, and this is easy to misdiagnose. Third, background subtraction with GMM does not suit footage from a moving camera even with motion compensation, because of reference drift and because it cannot tell fish motion from coral sway or caustics.

VI. CONCLUSION AND FUTURE WORK

We presented a semi-automatic pipeline for annotating, detecting and tracking fish in underwater video, built and tested entirely on CPU hardware. Classical detectors struggled with a moving camera, weak contrast and crowded schools; ByteTrack and BoT-SORT showed ID errors or high cost; and the combination of a YOLO26n detector trained on corrected annotations with a lightweight Kalman tracker reached about 90% tracking accuracy on the first video and generalised reasonably to a second one. The main finding is that growing the in-domain dataset, including fish-free background frames, mattered much more than longer training.

Future work includes annotating more frames, especially for species and colour patterns that are under-represented, such as high-contrast yellow-finned fish that are currently detected only partly (fin or tail). The existing boxes should be audited for consistency, since partial-body labels were seen for at least one species. The loss was still falling, so training should be resumed from the last checkpoint. Optical-flow-based frame interpolation could replace simple frame duplication, with attention to the motion blur and low-texture artefacts seen in early tests.

Finally, a dense-trained deep detector on a larger annotated set is the recommended path to production-level accuracy.

ACKNOWLEDGMENT

This work was carried out as part of an internship at the Aazhi Research Laboratory, Department of Electronics and Communication Engineering, National Institute of Technical Teachers Training and Research (NITTTR), Chennai. The authors gratefully acknowledge the support of NITTTR Chennai.

REFERENCES

[1] R. B. Fisher, Y.-H. Chen-Burger, D. Giordano, L. Hardman, and F.-P. Lin, Eds., *Fish4Knowledge: Collecting and Analyzing Massive Coral Reef Fish Video Data*. Cham, Switzerland: Springer, 2016.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2016, pp. 3464–3468.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[5] Y. Zhang *et al*., "ByteTrack: Multi-object tracking by associating every detection box," in *Proc. Eur. Conf. Comput. Vis. (*ECCV*)*, 2022, pp. 1–21.

[6] Roboflow, Inc., "Roboflow: Computer vision tools for annotation and model training." [Online]. Available: Roboflow