Fish Motion Estimation Using ML-based Relative Depth Estimation and Multi-Object Tracking

Lintao Fang
Fraunhofer IGD-R
Germany 18059, Rostock
fang.lintao@igd-r.fraunhofer.de

Mohamad Albadawi
mohamedbadawi9@gmail.com

Tim Dolereit
Fraunhofer IGD-R
Germany 18059, Rostock
tim.dolereit@igd-r.fraunhofer.de

Arjan Kuijper
Fraunhofer IGD
Germany 64283, Darmstadt
arjan.kuijper@igd.fraunhofer.de

Matthias Vahl
Fraunhofer IGD-R
Germany 18059, Rostock
matthias.vahl@igd-r.fraunhofer.de

ABSTRACT

Fish motion is a very important indicator of various health conditions of fish swarms in the fish farming industry. Many researchers have successfully analyzed fish motion information with the help of special sensors or computer vision, but their research results were either limited to few robotic fishes for ground-truth reasons or restricted to 2D space. Therefore, there is still a lack of methods that can accurately estimate the motion of a real fish swarm in 3D space. Here we present our Fish Motion Estimation (FME) algorithm that uses multi-object tracking, monocular depth estimation, and our novel post-processing approach to estimate fish motion in the world coordinate system. Our results show that the estimated fish motion approximates the ground truth very well and the achieved accuracy of 81.0% is sufficient for the use case of fish monitoring in fish farms.

Keywords
fish activity index, multi-object tracking, absolute depth map reconstruction, post-processing approach, motion estimation, fish swarm

1 INTRODUCTION

The level of fish activity serves as an important indicator in fish farming, providing biologists with insights into the condition of a fish swarm, such as hunger or sickness. For example, reduced activity is often observed in hungry fish, leading to a noticeable decline in their overall speed. Summarizing the motion characteristics of a fish swarm is one of many approaches to reflect their entire activity level. Some researchers in recent years have used computer vision [HZL+20, WML+21] or Doppler-based technique [HFPA20, HFPA19, HFU+22] to estimate real fish speed, but they cannot evaluate the error of their method because of the lack of real fish speed as the reference. Other researchers [WWX15, WLZ+16] found robotic fishes with special sensors could be an alternative because these sensors can provide the ground truth as a reference, but their experimental results are limited to very few fishes rather than fish swarms. Overall, these attempts consistently fail to reflect the true motion information of fish swarms in 3D space, leading to low persuasiveness and reliability.

In our work, we address the aforementioned limitations in two steps. First, it is technically difficult to obtain the real speed of a fish swarm in the real world, so we use the Unity game engine [JBT+18] to simulate different fish swarm scenes in the real world. By undertaking this approach, we can not only replicate real-world fish swarm scenarios but also utilize Unity to generate accurate fish swarm motion data and the ground truth. Second, we propose a novel algorithm as shown in Fig. 1 to robustly estimate fish motion in a simple, reproducible, and inexpensive manner. This algorithm consists of three main components, a multi-object tracking module, a monocular depth (mono-depth) estimation module, and a post-processing module. In our experiments, our unique post-processing method proved to be effective in overcoming the significant challenges caused by complex fish swimming motion in 3D space. The estimated average speed of the fish swarm is located in the world coordinate system, so our result can reflect the real state of fish motion. We use this average speed of the fish swarm as an index to represent the overall activ-

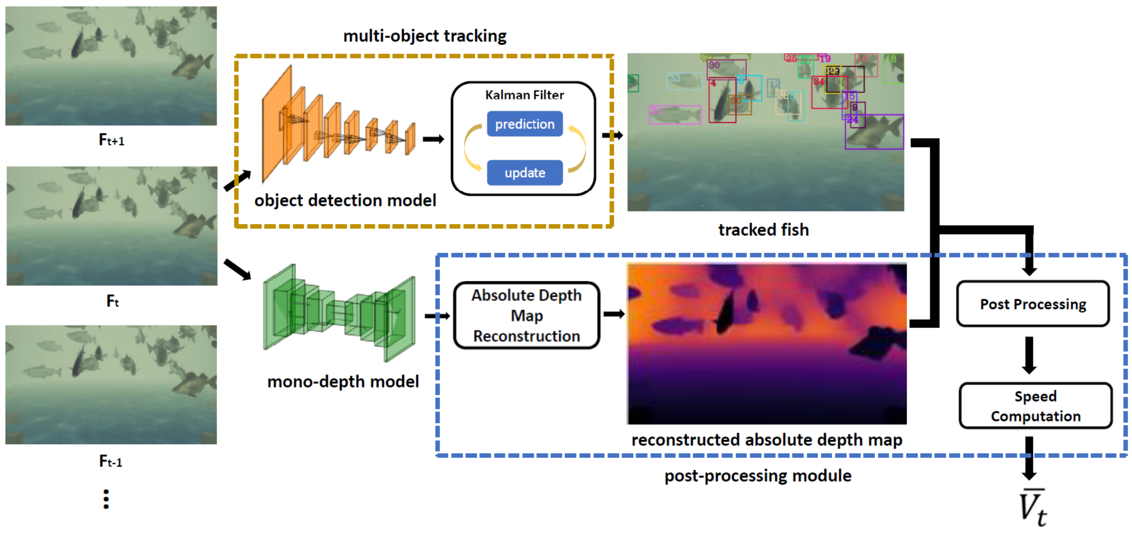

Figure 1: The overview of our Fish Motion Estimation (FME) algorithm. In a given frame F_t, we employ an object detector and Kalman Filter [Kal60] for fish tracking, assigning a unique ID to each fish. Simultaneously, a mono-depth model processes this image F_t and estimates the relative inverse depth map. From this, we reconstruct the absolute depth map. Afterward, we utilize our post-processing approach to process the tracking result and absolute depth map, and further we use this processing result to compute the fish swarm's average speed $\overline{V_t}$ in the world coordinate system.

ity level of the fish swarm. Although we study our Fish Motion Estimation (FME) algorithm in a synthetic environment, we believe that our approach can be easily and effectively transferred to the real world. Therefore, all experimental steps in the synthetic environment remain strictly consistent with those in the real world.

We structure this paper as follows. In Section 2, we summarize the recent approaches for fish motion estimation. In Section 3, we describe our approach to absolute depth map reconstruction, the post-processing approach, and speed computation in detail. In Section 4, we show the detailed experiment results in multi-object tracking, mono-depth estimation, and fish motion estimation. In Section 5, we summarize our work and present the future plan to improve our fish motion estimation algorithm.

Our main contributions include:

- Introduction of a new algorithm that accurately estimates how fishes move in 3D space.
- Creation of a novel post-processing method to handle the challenges posed by the way fishes swim, making our approach stand out.
- A step forward by looking at how fishes move not just in 2D but also in 3D space, offering a more complete picture.

2 RELATED WORK

Fish studies in the early days relied on video systems to analyze fish swimming movements from video recordings, such as FICASS [PSW+97]. Later on, some researchers used camera-based computer-controlled devices to perform real-time processing of fish kinematic information, including swimming acceleration and velocity [WZ07, CXG+09]. However, their research results are only based on few fishes. Other researchers use the Doppler principle-based technique to measure the motion information of fish swarms in sea cages [HFPA20, HFPA19, HFU+22]. Unlike the Doppler-based technique, some researchers used CNN-based methods to continuously track multiple fishes and further use the tracking results to estimate the fish swimming speed [LXH+21, BPPLAM23]. However, their experimental findings are less reliable as they cannot compare their results with ground truth, and their works are limited in 2D space. Zhang et al. [ZZH+23] utilized a binocular camera to project the 2D tracking space into 3D space and compare their estimated fish swimming speed with their ground truth in that space, but their ground truth is derived from pixel coordinates rather than from the real world.

To enhance the reliability of experimental results, subsequent researchers replaced the real fish with the controllable robotic fish because they can obtain the real swimming speed of the robotic fish as the ground truth. However, these robot fish-based methods require additional assistance, such as the optimal information fusion decentralized Kalman filter algorithm [WWX15], pressure sensor [WLZ+16, ZZX14], and the deep reinforcement learning controller [DSAR24, DNSR23]. Nevertheless, their experimental subjects only involve very few robotic fishes rather than fish swarms.

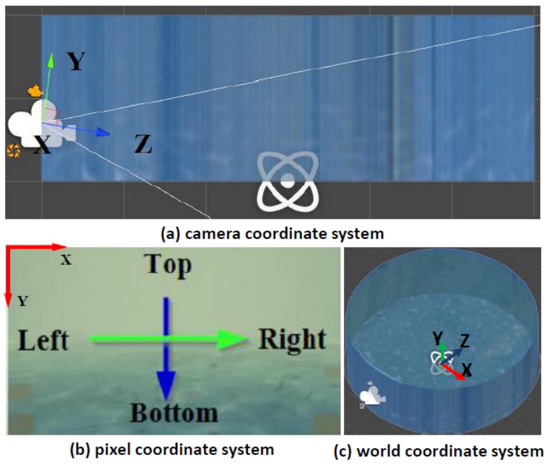
Y
X
Z
(a) camera coordinate system
X
Y
Top
Left
Right
Bottom
(b) pixel coordinate system
Y Z
X
(c) world coordinate system

Figure 2: Three types of coordinate systems. (b) The arrows represent the fish's swimming directions.

To overcome these dilemmas, we use 3D fish models and the Unity game engine to simulate real-world fish swarm scenes. Under these settings, we can obtain real fish speed in 3D space as the ground truth. Since our synthetic fish scenes are in the world coordinate system, our work extends fish motion estimation from 2D to 3D space.

3 METHOD

3.1 Absolute Depth Map Reconstruction

To estimate the fish motion information in World Coordinate System (WCS) shown in Fig. 2 (c), we first need to acquire fish position information in WCS. As shown in Fig. 1, we use MiDaS Hybrid [RLH+22] as the mono-depth model to estimate relative inverse depth R_{inv}, and use an object detector to estimate fish position in the Pixel Coordinate System (PCS) shown in Fig. 2 (b). The depth R_{inv} predicted by MiDaS Hybrid is scaled and shifted to a unit scale $s = 1$ and zero translation $t = 0$, so it loses the absolute depth information of the objects in the scene and only presents the relative positional relationship between objects. However, we need the absolute depth information to convert fish position information from PCS to that in WCS. To solve this problem, we use an optimized transform algorithm as shown in Equation (1) to convert R_{inv} to absolute depth D_{abs}.

$$s^*, t^* \leftarrow argmin\ L_2(\mathbf{D}_{dv}, \mathbf{R}_{dv}) \tag{1}$$

D_{dv} is the known absolute depth values in the scene, while R_{dv} is the relative inverse depth values in the depth map R_{inv}. This algorithm requires at least two known absolute depth values D_{dv} and two relative inverse depth values R_{dv} to solve s^* and t^*.

We place ten reference objects in the fish pond to provide the known absolute depth values from the fish scene. We take the center point on the front surface from each reference object as the reference point p_i. We locate the coordinate of each reference point $p_i = (w_x^i, w_y^i, w_z^i)$ in WCS. Since Unity game engine can offer camera extrinsic parameter $\mathbf{E}$ and intrinsic parameter $\mathbf{I}$, we can convert these reference points in WCS to those in the Camera Coordinate System (CCS) shown in Fig. 2 (a). Afterward, we collect the absolute depth values D_{dv} from these reference points. We continue to convert these reference points in CCS to those in PCS so that we use the pixel coordinates of these reference points to extract R_{dv} from the relative inverse depth map R_{inv}. In the end, we use the known absolute depth values D_{dv} and relative inverse depth values R_{dv} to solve the optimal solution s^* and t^*. With s^* and t^*, we can reconstruct the absolute depth D_{abs} by using following the Equation (2).

$$\mathbf{D}_{abs} \leftarrow \mathbf{R}_{inv} \times s^* + t^* \tag{2}$$

3.2 Post Processing

Fish swimming movements are random, unpredictable, and easily influenced by their states as well as the surroundings [RPM22, HZL+20]. These special features eventually cause two problems, which are the fluctuation in the position of the BBox center point and the abrupt jump in the depth value at that point. These two problems will cause large errors in speed computation because we estimate fish speed based on the BBox center points.

We assume that the location of a fish at the current frame f is based on its location at previous frames. Therefore, our idea is to summarize the past location information of a fish and use it to update its location information in the current frame f. We use the Exponentially Weighted Moving Average (EWMA) to achieve it.

3.2.1 Fluctuation problem of BBox center point

Due to the various fish swimming postures or fish-tail swinging movements, these factors could cause the BBox dimensions to change abnormally between successive sequences. For example, the width of BBox may suddenly become larger or smaller in two successive frames $t-1$ and t when a fish is swimming. This leads to the fluctuation problem on the BBox center point, and it can negatively affect the estimation results of fish speed. To overcome this problem, we create two detectors as described in Equation (3) to monitor abnormal changes in the BBox dimensions. We use the average variation **Avg(.)** of the BBox dimensions in the past 30 frames as the reference to determine if the current change in the BBox dimension is abnormal.

$$\begin{aligned} detector_w &= \mathbf{abs}(w_t - w_{t-1}) > \mathbf{Avg}(\mathbf{D}_w, Len(\mathbf{D}_w)) \\ detector_h &= \mathbf{abs}(h_t - h_{t-1}) > \mathbf{Avg}(\mathbf{D}_h, Len(\mathbf{D}_h)). \end{aligned} \tag{3}$$

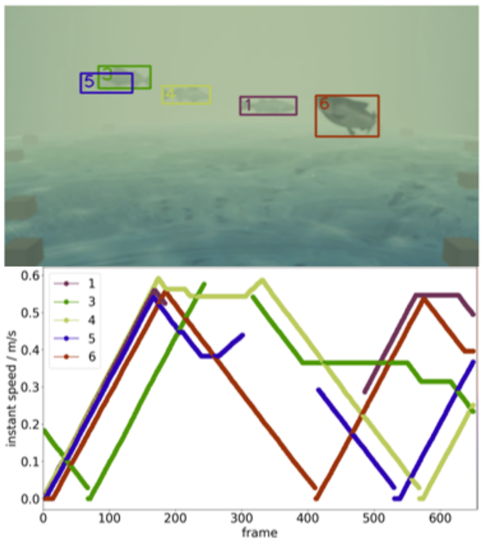

Figure 3: Annotation. The real-time speed of each fish in *WCS*.

$\mathbf{D}_w = \{(w_i - w_{i-1})\}_{i=t-28}^{t-1}$ is the list containing the difference between the BBox width w in every two adjacent frames, while the $\mathbf{D}_h = \{(h_i - h_{i-1})\}_{i=t-28}^{t-1}$ represents the difference in BBox height h. If we detect an abnormal change in the BBox dimensions at frame t when a fish swims, we collect its BBox dimensions in the previous 5 frames. Afterward, we use EWMA to update the BBox dimensions at the current frame t and further compute the BBox center point $c_t = (c_t^x, c_t^y)$.

3.2.2 Jump problem on the depth value

Abrupt changes in depth value at the BBox center points occur mainly when a fish is partially occluded by others during fish interactions. This occlusion problem causes the fish BBox center point to jump onto other fishes in the next frame making the depth value inaccurate. To tackle this problem, we first create a detector as shown in Equation (4) to monitor each fish and detect if its depth value at the BBox center point jumps. We assume that the change in depth value for each fish in successive frames is smooth when a fish swims alone. $\mathbf{abs}(d_t - d_{t-1})$ is the absolute depth difference in successive frames. We experimentally found that $\delta = 0.6$ m delivers good results. Together with the BBox center point $c_t = (c_t^x, c_t^y)$ at current frame t, the absolute depth map $\mathbf{D}_{abs}^t$ and the set of depth values in past 30 frames $\mathbf{Dep} = \{d_i^*\}_{i=t-29}^{t-1}$, we use EWMA to update the depth value at the current frame t. Eventually, we obtain the updated absolute depth value d_t^*.

$$detector_{dep} = \mathbf{abs}(d_t - d_{t-1}) \geq \delta. \tag{4}$$

3.3 Speed computation

In order to estimate the average speed of the fish swarm, we start with a single fish. We use the BBox center points $\mathbf{C} = \{(c_i^x, c_i^y)\}_{i=1}^t$ and the absolute depth values $\mathbf{Dep} = \{d_i^*\}_{i=1}^t$ to firstly project BBox center points $\mathbf{C}$ in PCS to WCS, and then we compute fish speed $\mathbf{V} = \{v_i\}_{i=1}^t$. The time interval is $1/30$ seconds, so $fps = 30$. Next, we still use the above computation steps to calculate the speed for each fish in a fish swarm. With our multi-object tracker, we can identify each fish in the next frames. For each frame, we use Equation (5) to compute the average speed $\bar{v}_t$ of all tracked fishes. Here N is the total number of fishes at the frame t. After traversing all frames T, we obtain the average speed $\bar{\mathbf{V}} = \{\bar{v}_t\}_{t=1}^T$ as the final result.

$$\bar{v}_t = \frac{1}{N} \sum_{o=1}^{N} v_t^o, \tag{5}$$

Hyperparameter	Symbol	Search Space
Kernel Size	k	$[3\times3, 5\times5, 7\times7]$
Detection Thres	T_d	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Tracking Thres	T_t	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Fusion Thres	T_f	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Untracked Thres	T_{unt}	$[30\%, 40\%, 50\%, 60\%, 70\%]$
Unconfirmed Thres	T_{unc}	$[30\%, 40\%, 50\%, 60\%, 70\%]$
IoU Thres	T_{iou}	$[30\%, 40\%, 50\%, 60\%, 70\%]$

Table 1: Grid search space for hyper-parameters

4 EXPERIMENTS

4.1 Synthetic Fish Dataset

There are some fish datasets available for different computer vision tasks, such as [UKT20] and [SLK+20], but none of them provides fish motion information. Therefore, we use Unity game engine [JBT+18] to design our synthetic fish dataset and automatically generate annotation as shown in Fig. 3. This synthetic dataset contains 163 videos which have 89901 images in total. We split this dataset into training dataset with 99 videos, validation dataset with 32 videos, and testing dataset with 32 videos. The frame per second in this dataset is 30 fps.

Unity captures the ground truth speeds based on the gravity center of the 3D fish model. It is very difficult to visually determine the position of the gravity center of a swimming fish, so we choose to approximate the ground truth speed by the speed of the BBox center point. This is expected to introduce inherent errors, primarily stemming from two distinct factors. Firstly, the likelihood of the gravity center aligning precisely with the center point of the BBox diminishes when a fish is

Trackers	k	T_d	T_t	T_f	T_{unt}	T_{unc}	T_{iou}
T_{hrnet}	3×3	30%	70%	70%	70%	30%	–
T_{retina}	–	30%	50%	–	70%	30%	50%
$T_{fasterrcnn}$	–	50%	50%	–	70%	40%	50%

Table 2: Grid search result for multi-object trackors

Trackers	IDF1 ↑	MOTA ↑	MOTP ↑	fps ↑
T_{hrnet}	80.1%	81.1%	**87.4%**	16.7
T_{retina}	79.4%	80.5%	86.7%	20.39
$T_{fasterrcnn}$	**82.4%**	**81.8%**	**87.4%**	**21.15**

Table 3: Multi-object tracking results

navigating through three-dimensional space. Secondly, the depth value at the BBox center point consistently corresponds to the outer surface of the fish, whereas the gravity center resides within the fish. Consequently, the depth value at the BBox center point is almost always smaller than the value at the gravity center.

4.2 Multi-Object Tracking

We adopt the tracking-by-detection approach to achieve multiple fish tracking in our synthetic fish swarm scenes. We prepare three detector-based multi-object trackers, including HRNet-based tracker T_{hrnet}, RetinaNet-based tracker T_{retina}, and FasterRCNN-based tracker $T_{fasterrcnn}$. We use the same training strategy to train these object detectors. The batch size is set to 10. We optimize these models with the Adam optimizer [KB14] for 40 epochs. The learning rate γ is $1.25e^{-4}$, and it is decayed by half for every 10 epochs. After the training stage, we use each well-trained object detector and Kalman Filter [Kal60] to compose a multi-object tracker. Since Kalman Filter involves some hyperparameters, we create a search space for each tracking hyperparameter shown in Table 1 and use the grid search approach to find the optimal tracking parameter for each tracker. We show the result in Table 2.

We use the evaluation metrics that are commonly adapted in the community to measure tracking performance. The tracking result is shown in Table 3. T_{hrnet} has comparable overall performance with T_{retina}, but $T_{fasterrcnn}$ outperforms other trackers in tracking performance and inference speed. Therefore, we will use the FasterRCNN-based tracker to achieve multiple fish tracking in the following experiments.

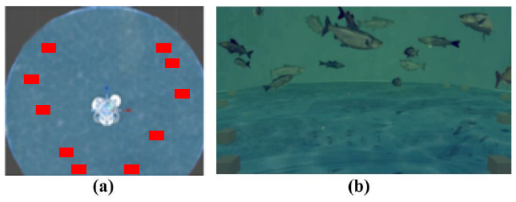

Figure 4: Random layout of reference objects. (a) The red boxes are the positions of reference objects. (b) The reference objects from the front view.

4.3 Absolute Depth Map Reconstruction

Since it is technically difficult to generate precise depth maps as ground truth in underwater environments, we cannot train any mono-depth models in the real world. Similarly, we do not train any mono-depth models on our synthetic datasets, and we use an already well-trained model MiDas [RLH+22] with high generalization ability.

4.3.1 *Transformation Algorithms*

We choose four open-source mono-depth variants based on MiDaS as our experimental candidates, and they are MiDaS Hybrid [RLH+22], MiDaS Large [RLH+22], BoostingMiDaS [MDM+21] and TCMiDaS [LLZ+21]. We use four Machine Learning (ML) algorithms to reconstruct absolute depth maps from the relative inverse depth maps, including Linear Regression (LR), second-order Polynomial Regression (PR), Decision Tree (DT), and KNN. There are two primary reasons for this operation. Firstly, these models lack the capability to directly predict absolute depth maps. Secondly, [RLH+22, MDM+21, LLZ+21] do not put forth any default transformation algorithms

Reconstruction Algorithms		$RMSE/m$ ↓	&1.25 ↑	&1.25^2 ↑
mono-depth models	ML			
MiDaS Hybrid	LR	**0.625**	**0.73**	**0.94**
	PR	27.36	0.72	0.925
	DT	0.66	0.695	0.915
	KNN	0.655	0.7	0.92
MiDaS Large	LR	0.93	0.58	0.825
	PR	42.915	**0.605**	0.84
	DT	0.79	0.575	**0.855**
	KNN	**0.785**	0.59	0.85
Boosting MiDaS	LR	0.725	**0.675**	**0.9**
	PR	118.3	0.66	0.885
	DT	0.71	0.665	**0.9**
	KNN	**0.7**	0.67	**0.9**
TCMiDaS	LR	1.675	**0.355**	0.535
	PR	2934.985	**0.44**	**0.62**
	DT	1.485	0.38	0.6
	KNN	**1.415**	0.375	0.61

Table 4: Comparison of different absolute depth map reconstruction algorithms

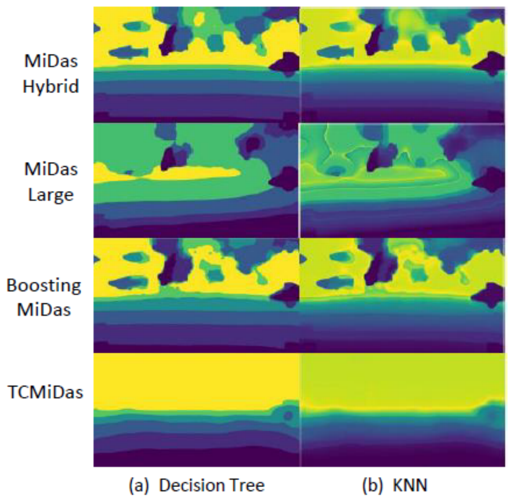

Figure 5: Examples of reconstructed absolute depth maps

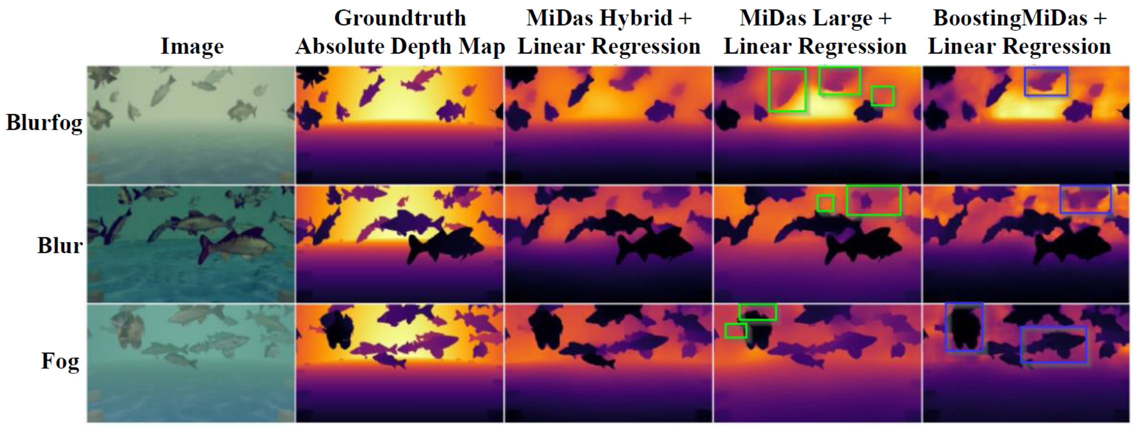
Image
Groundtruth Absolute Depth Map
MiDas Hybrid + Linear Regression
MiDas Large + Linear Regression
BoostingMiDas + Linear Regression
Blurfog
Blur
Fog

Figure 6: Examples of the absolute depth maps reconstructed by linear regression in different fish scenes. **The green rectangle** marks the ignored fish. **The blue rectangle** marks fish with the same depth values in a small fish swarm.

that can effectively convert a relative inverse depth map into an absolute depth map. We use the Root Mean Squared Error (RMSE) to measure the deviation between the estimated depth values and the ground truth. We also use &1.25 and &1.25^2 to evaluate their precision. Since absolute depth map reconstruction requires some known depth values from the scene, we tried different layouts of the reference objects and came up with the one that performed best, as shown in Fig. 4 (a).

A depth estimation algorithm is deemed effective if it achieves continuous values in the predicted depth map without introducing layering effects, while also maintaining good metrics for both the background and foreground objects. We combine four mono-depth variants and four ML algorithms and evaluate their absolute depth map reconstruction results. The result is shown in Table 4. We find that polynomial regression ends up with a very large error in RMSE. This primarily stems from the x^2 term in polynomial regression, which tends to magnify large depth values during the depth map reconstruction process. Thus, it is unsafe to use polynomial regression to conduct depth map transformation. As shown in Fig. 5, the decision tree and KNN tend to reconstruct absolute depth maps with obvious layering effects, so it is not acceptable in our case. Additionally, the reconstructed depth maps based on TCMiDaS lose most of the depth information. It is probably because the training strategy in Li et al. [LLZ+21] does not maintain the original generalization capability of MiDas. Thus, we will not consider this model in our work. Unlike other ML algorithms, linear regression maintains the most balanced performance in all metrics and with all Midas variants. Therefore, linear regression is the most suitable transformation algorithm.

We further compare the absolute depth maps reconstructed by linear regression in different fish scenes. As shown in Fig. 6, MiDaS Large tends to miss some fishes that are relatively far away in the scene when estimating depth maps. BoostingMiDaS can provide depth information for all fishes, but it tends to predict the same depth values for neighboring fishes forming a group in the image. The reason may be that the fish's appearance and the boundaries between these fishes become blurred in these scenes, so BoostingMiDaS incorrectly recognizes their contextual cues as the same. Such a situation makes it difficult for BoostingMidas to distinguish multiple fishes with blurred boundaries. MiDaS Hybrid and linear regression do not have the problems mentioned above, so we will use MiDaS Hybrid and linear regression to reconstruct absolute depth maps for our fish scenes.

4.3.2 *Layout of Reference Objects*

The spatial layout of reference objects is another important factor in absolute depth map reconstruction. As shown in Fig. 7, we prepare seven different spatial layouts and restrict the size of all reference objects to $0.1 \times 0.1 \times 0.1m^3$. We assume that the reference objects in this size are too small to affect fish swimming movements.

The result is shown in Table 5. We find that the increasing number of reference objects from layouts A to

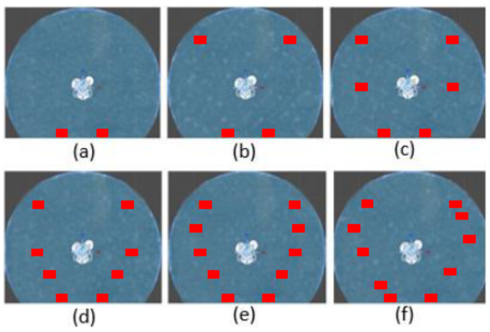

Figure 7: Different layouts of reference objects. The red bboxes are the positions of reference objects

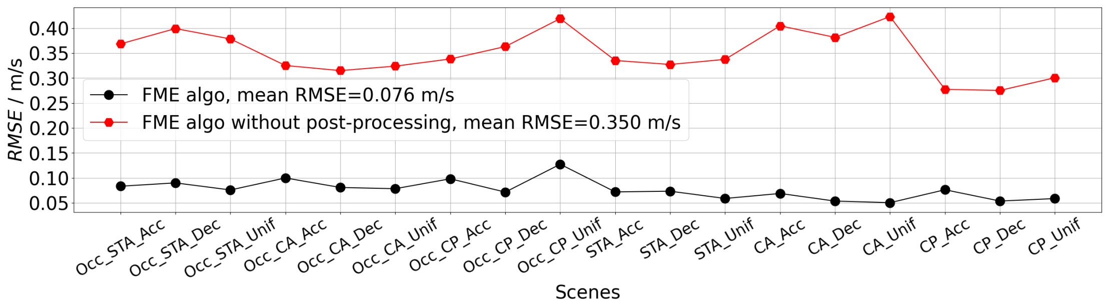

Figure 8: Post-processing performance. The **Occ** means occlusion.

D improves the overall accuracy in absolute depth map reconstruction and decreases errors in reconstructed absolute depth values. The reason is that placing more reference objects in the depth direction of the fish pond can provide more depth information about the pond. However, as shown in Table 5, placing two more objects in layout E shown in Fig. 7 ends up with a very similar evaluation result to that of layout D. This result indicates that placing more objects is less helpful in improving the accuracy of absolute depth map reconstruction. To solve this problem, we created layout F by slightly shifting each reference object in different directions by a small distance. In Table 5, the overall evaluation result in layout F delivers better results than other layouts. Compared to layout E, layout F uses the same number of reference objects to obtain a more complete depth range in the fish pond. Therefore, we will arrange reference objects in our fish ponds according to layout F.

4.4 Post Processing

To investigate the effectiveness of our post-processing method against the fluctuation problem of BBox center point and the jump problem on depth value, we created simple scenes where a fish swims in pre-designed movements. These scenes include three types of motion and three types of swimming trajectories. The three types of motion include acceleration (Acc), deceleration (Dec), and uniform (Unif). The three types of trajectories include the Coordinate Axis direction (CA), the Coordinate Plane direction (CP), and Spatial Turn-Around movements (STA). In detail, the CA direction includes directions in $\overrightarrow{d} = (1,0,0)$, $\overrightarrow{d} = (0,1,0)$, and

Layout	$RMSE/m \downarrow$	$\&1.25 \uparrow$	$\&1.25^2 \uparrow$
A	1.84	0.1	0.525
B	0.65	0.68	0.89
C	0.59	0.735	0.93
D	0.565	0.755	0.95
E	0.57	0.75	**0.955**
F	**0.55**	**0.77**	0.95

Table 5: Absolute depth map reconstruction in different layouts

$\overrightarrow{d} = (0,0,1)$, while the CP direction includes directions in $\overrightarrow{d} = (1,1,0)$, $\overrightarrow{d} = (1,0,1)$, and $\overrightarrow{d} = (0,1,1)$. Since a fish is either occluded or not occluded when swimming in a fish swarm, we divide these scenes into the occluded and not-occluded ones. In total, we have 18 different scenes. Also, we feed the ground truth BBox and depth map to our post-processing approach rather than the tracking result and reconstructed absolute depth maps as shown in Fig. 1.

As shown in Fig. 8, our FME algorithm can produce better results in occluded and non-occluded scenes than FME algorithm without the post-processing approach. Our post-processing approach reduces $\nabla RMSE$ by $0.274 m/s$. This result implies that our post-processing approach effectively addresses the disturbance caused by fish swimming movement. Following, two examples are given to demonstrate the improvements in fish motion estimation.

Since fish swimming movements always cause the drift problem on BBox center point, we make a single fish swimming in the direction $\overrightarrow{d} = (1,0,0)$ as shown in Fig. 9 in order to investigate the influence of this problem on the fish motion. The fish swimming along this trajectory can always be at the same distance from the

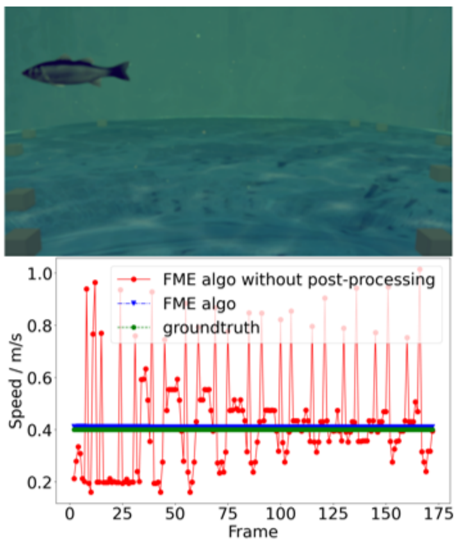

Figure 9: Post processing on BBox

camera. As shown in Fig. 9, the estimated result based on our post-processing approach remains exactly the same as the ground truth while the method without the post-processing generates a very fluctuating result. This comparison indicates that our post-processing approach can overcome the drift problem on BBox center points. When a fish is swimming behind another fish as shown in Fig. 10, it causes the depth value at the BBox center point to abruptly jump in two consecutive frames. However, our post-processing approach can overcome this problem and maintain smooth changes in depth values on successive frames.

4.5 Fish motion estimation

We have prepared two types of fish scenes, including the single-fish scene and the fish swarm. Our initial focus is on the single-fish scene. This choice allows us to fully isolate the effects of interaction between fishes and concentrate on assessing the influence of the spatial swimming movements of an individual fish on our FME algorithm. We compare our FME algorithms based on four different input sources that include Ground Truth BBox (GTBBox), Ground Truth depth (GTDepth), Predicted BBox (PredBBox), and Estimated absolute Depth map (EstDepth). We consider the result from the algorithm based on GTBBox&GTDepth as the benchmark.

We utilize RMSE to evaluate the deviation between the estimated speed and the ground truth speed. We also use distance correlation (DistCorr) to quantify the similarity of tendency between them. Unlike the Spearman correlation and the Pearson correlation, DistCorr can measure the nonlinear association between two variables with non-monotonicity, and it does not impose any restrictions on the distribution of these two variables.

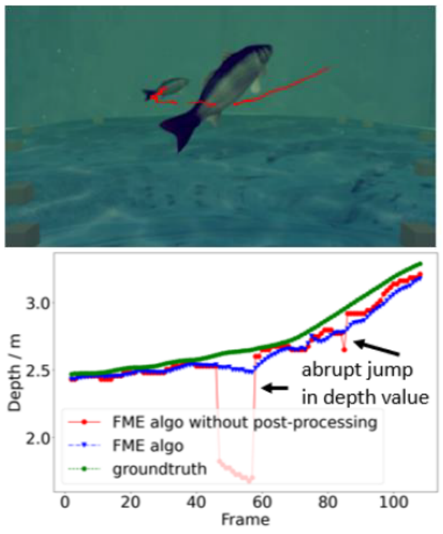

Figure 10: Post processing on depth. The red line on the top figure represents the fish swimming trajectory.

4.5.1 *Single-Fish Scene*

As shown in Table 6, the benchmark result ends up with an error of $RMSE = 0.05m/s$, while the correlation reaches up to $DistCorr = 92.0\%$. Since we only use ground truth depth maps and BBox to estimate fish speed, we think this error mainly comes from the inherent error as we discussed in Section 4.1 and our post-processing approach.

The estimation result in group C is very close to the benchmark, giving an error of $RMSE = 0.06m/s$ and the correlation of $DistCorr = 92.0\%$. The reason for this difference is that BBoxes predicted by our tracking algorithm deviate a bit from the ground truth BBox, so it leads to a localization error in the fish positions. However, its performance remains very close to the benchmark as shown in Fig. 11 (b). We notice that the result in Group B is worse than that in the benchmark. This is mainly because the reconstructed absolute depth map provides less accurate depth values for fish speed computation. In group D, the combination of our tracking and absolute depth reconstruction algorithms can generate nearly the same performance as that in group B. It implies that our absolute depth estimation algorithm is the main error source in single-fish motion estimation. However, the estimated speed shown in Fig. 11 (c) approximates the ground truth well enough. Thus, we believe that our algorithm can robustly estimate fish motion in single-fish scenes with an error of $RMSE = 0.08m/s$ and the correlation of $DistCorr = 89.0\%$.

4.5.2 *Fish Swarm Scene*

As shown in Table 7, we notice that the benchmark result has an error of $RMSE = 0.03m/s$, but the correlation between the estimated average speed and the ground truth is $DistCorr = 92.0\%$. The estimated result in Group D has an error of $RMSE = 0.06m/s$. By comparing with the benchmark, our tracking and absolute depth map reconstruction algorithms together lead to an increasing error in the estimated average speed by $\nabla RMSE = 0.03m/s$. Also, its correlation of $DistCorr = 81.0\%$ differs from the benchmark by $\nabla DistCorr = 11.0\%$. The reasons are two-fold. First, fish swarm scenes make a more difficult task for our tracker because of occlusion, which makes it more difficult to continuously and stably track a fish with the same tracking ID, and increases the chances of losing some tracked targets or ID switches while tracking

Group	Input Source	$RMSE(m/s)\downarrow$	$DistCorr\uparrow$
A	Benchmark	**0.05**	**92.0%**
B	GTBBox&EstDepth	0.08	90.0%
C	PredBBox&GTDepth	0.06	**92.0%**
D	PredBBox&EstDepth	0.08	89.0%

Table 6: Fish motion estimation in single-fish scenes

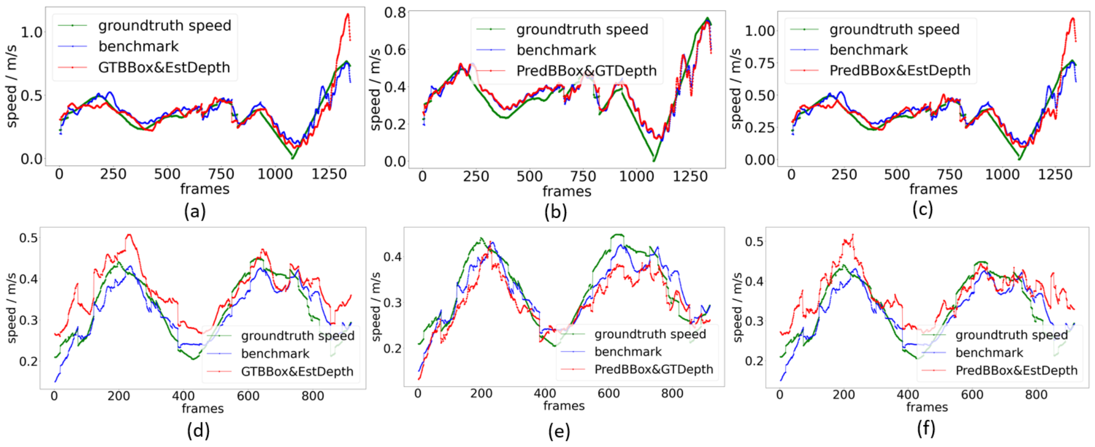

Figure 11: Fish motion estimation. (a)-(c): single fish scenes. (d)-(f): fish swarm scenes.

many fishes. This problem is highly dependent on the complexity of fish interaction in a fish swarm. The imperfection of our tracking algorithm on motion estimation becomes more obvious in fish swarm scenes. Second, fish swarm scenes increase the difficulty of reconstructing accurate absolute depth information for the fish. These two aspects eventually lead to the deviation in fish speed computation. However, the estimated speed curve shown in Fig. 11 (f) closely mirrors the trend of the ground truth curve. Thus, we believe that our tracking model and mono-depth estimation algorithm can adequately estimate fish motion in fish swarm scenes with $RMSE = 0.06m/s$ and $DistCorr = 81.0\%$. Our biologist partners believe that this error is acceptable as long as the estimated speed curve is close to the ground truth curve.

Additionally, we adopt a 95% confidence interval to measure the error range of the average speed in a fish swarm at a given frame f computed by our FME algorithm, and the range is $[-0.167m/s, 0.1m/s]$. This error range is measured under the frame rate of 30 fps.

5 CONCLUSION

In our research, we tackled challenges in fish motion studies by addressing the lack of ground truth information, and the complexity of fish swarm scenes. We created synthetic fish data and introduced a novel fish motion estimation algorithm, incorporating multi-object tracking, a mono-depth model, and an innovative post-processing approach for 3D motion estimation. Our research successfully extends fish studies from 2D to 3D space. Our experiments show that our post-processing method significantly reduces errors in fish speed computation, and our FME algorithm can estimate adequately accurate fish motion in different fish swarm scenes.

Group	Input Source	$RMSE(m/s)\downarrow$	$DistCorr\uparrow$
A	Benchmark	**0.03**	**92.0%**
B	GTBBox&EstDepth	0.05	88.0%
C	PredBBox&GTDepth	0.04	**92.0%**
D	PredBBox&EstDepth	0.06	81.0%

Table 7: Fish motion estimation in fish swarm scenes

Our work demonstrates the feasibility of accurately computing fish motion using a multi-object tracker and a mono-depth model. Looking ahead, we aim to design a neural network model for direct fish speed prediction.

6 ACKNOWLEDGMENT

This project was funded by the Federal Ministry of Education and Research under the funding code 03ZU1107MB.

7 REFERENCES

[BPPLAM23] Faezeh Behzadi Pour, Lorena Parra, Jaime Lloret, and Saman Abdanan Mehdizadeh. Measuring and evaluating the speed and the physical characteristics of fishes based on video processing. *Water*, 15(11):2138, 2023.

[CXG$^+$09] Jiujun Chen, Gang Xiao, Fei Gao, Hongbin Zhou, and Xiaofang Ying. Vision-based perceptive framework for fish motion. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4. IEEE, 2009.

[DNSR23] Palmani Duraisamy, Manigandan Nagarajan Santhanakrishnan, and Amirtharajan Rengarajan. Design of deep reinforcement learning controller through data-assisted model for robotic fish speed tracking. *Journal of Bionic Engineering*, 20(3):953–966, 2023.

[DSAR24] Palmani Duraisamy, Manigandan Nagarajan Santhanakrishnan, Rengarajan Amirtharajan, and Sudha Ramasamy. Real-time implementation of deep reinforcement learning controller for speed tracking of robotic fish through data-assisted modeling. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 238(2):572–585, 2024.

[HFPA19] Waseem Hassan, Martin Føre, Magnus Oshaug Pedersen, and Jo Arve Alfredsen. A novel doppler based speed measurement technique for individual free-ranging fish. In *2019 IEEE sensors*, pages 1–4. IEEE, 2019.

[HFPA20] Waseem Hassan, Martin Føre, Magnus Oshaug Pedersen, and Jo Arve Alfredsen. A new method for measuring free-ranging fish swimming speed in commercial marine farms using doppler principle. *IEEE Sensors Journal*, 20(17):10220–10227, 2020.

[HFU$^+$22] Waseem Hassan, Martin Føre, Henning A Urke, John B Ulvund, Eskil Bendiksen, and Jo A Alfredsen. New concept for measuring swimming speed of free-ranging fish using acoustic telemetry and doppler analysis. *biosystems engineering*, 220:103–113, 2022.

[HZL$^+$20] Fangfang Han, Junchao Zhu, Bin Liu, Baofeng Zhang, and Fuhua Xie. Fish shoals behavior detection based on convolutional neural network and spatiotemporal information. *IEEE Access*, 8:126907–126926, 2020.

[JBT$^+$18] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LLZ$^+$21] Siyuan Li, Yue Luo, Ye Zhu, Xun Zhao, Yu Li, and Ying Shan. Enforcing temporal consistency in video depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1145–1154, 2021.

[LXH$^+$21] Xianghui Li, Xin Xia, Zhuhua Hu, Bing Han, and Yaochi Zhao. Intelligent detection of underwater fish speed characteristics based on deep learning. In *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, pages 182–189. IEEE, 2021.

[MDM$^+$21] S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9685–9694, 2021.

[PSW$^+$97] RJ Petrell, X Shi, RK Ward, A Naiberg, and CR Savage. Determining fish size and swimming speed in cages and tanks using simple video techniques. *Aquacultural Engineering*, 16(1-2):63–84, 1997.

[RLH$^+$22] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[RPM22] Stefan Reiser, Erick Cantu Perez, and Alexandra Meier. Automated detection of fish activity in recirculating aquaculture systems. 2022.

[SLK$^+$20] Alzayat Saleh, Issam H Laradji, Dmitry A Konovalov, Michael Bradley, David Vazquez, and Marcus Sheaves. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Scientific Reports*, 10(1):14671, 2020.

[UKT20] Oguzhan Ulucan, Diclehan Karakaya, and Mehmet Turkan. A large-scale dataset for fish segmentation and classification. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, 2020.

[WLZ$^+$16] Wei Wang, Yuan Li, Xingxing Zhang, Chen Wang, Shiming Chen, and Guangming Xie. Speed evaluation of a freely swimming robotic fish with an artificial lateral line. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4737–4742. IEEE, 2016.

[WML$^+$21] Guangxu Wang, Akhter Muhammad, Chang Liu, Ling Du, and Daoliang Li. Automatic recognition of fish behavior with a fusion of rgb and optical flow data based on deep learning. *Animals*, 11(10):2774, 2021.

[WWX15] Chengcai Wang, Wei Wang, and Guangming Xie. Speed estimation for robotic fish using onboard artificial lateral line and inertial measurement unit. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 285–290. IEEE, 2015.

[WZ07] GuanHao Wu and LiJiang Zeng. Video tracking method for three-dimensional measurement of a free-swimming fish. *Science in China Series G: Physics, Mechanics and Astronomy*, 50(6):779–786, 2007.

[ZZH$^+$23] Lanyue Zhang, Guilin Zhai, Bo Hu, Zhi Qiao, and Peizhen Zhang. Fish target detection and speed estimation method based on computer vision. In *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 1330–1336. IEEE, 2023.

[ZZX14] Qixin Zhu, Kun Zhong, and Guangming Xie. Speed estimation for robotic fish based on pressure sensor. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 2714–2718. IEEE, 2014.