electronics

MDPI

Article

Fish Monitoring from Low-Contrast Underwater Images

Nikos Petrellis [1,*], Georgios Keramidas [2], Christos P. Antonopoulos [1] and Nikolaos Voros [1]

[1] Electrical and Computer Engineering, University of Peloponnese, 263 34 Patras, Greece; ch.antonop@uop.gr (C.P.A.); voros@uop.gr (N.V.)
[2] Computer Science, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece; gkeramidas@csd.auth.gr
* Correspondence: npetrellis@uop.gr; Tel.: +30-2610-369-215

Abstract: A toolset supporting fish detection, orientation, tracking and especially morphological feature estimation with high speed and accuracy, is presented in this paper. It can be exploited in fish farms to automate everyday procedures including size measurement and optimal harvest time estimation, fish health assessment, quantification of feeding needs, etc. It can also be used in an open sea environment to monitor fish size, behavior and the population of various species. An efficient deep learning technique for fish detection is employed and adapted, while methods for fish tracking are also proposed. The fish orientation is classified in order to apply a shape alignment technique that is based on the Ensemble of Regression Trees machine learning method. Shape alignment allows the estimation of fish dimensions (length, height) and the localization of fish body parts of particular interest such as the eyes and gills. The proposed method can estimate the position of 18 landmarks with an accuracy of about 95% from low-contrast underwater images where the fish can be hardly distinguished from its background. Hardware and software acceleration techniques have been applied at the shape alignment process reducing the frame processing latency to less than 0.5 us on a general purpose computer and less than 16 ms on an embedded platform. As a case study, the developed system has been trained and tested with several Mediterranean fish species in the category of seabream. A large public dataset with low-resolution underwater videos and images has also been developed to test the proposed system under worst case conditions.

Keywords: fish monitoring; shape alignment; machine learning; ensemble of regression trees; shape orientation; morphological feature extraction; hardware acceleration

Citation: Petrellis, N.; Keramidas, G.; Antonopoulos, C.P.; Voros, N. Fish Monitoring from Low-Contrast Underwater Images. *Electronics* **2023**, *12*, 3338. https://doi.org/10.3390/electronics12153338

Academic Editor: Byung Cheol Song

Received: 10 June 2023
Revised: 25 July 2023
Accepted: 2 August 2023
Published: 4 August 2023

1. Introduction

Measuring fish morphological features and observing their behavior, are important tasks that have to be carried out daily in fish farms. The purpose of these tasks is to assess the growing conditions and the welfare of the fish, their health and feeding needs as well as to decide the optimal time for harvest. Morphological feature measurement includes the estimation of body dimensions and mass. Indicators of fish health include eye diameter and color, gill color as well as malformations in the shape. The prompt diagnosis of fish disease infections is necessary for lower cost treatment that will prevent the spread of a disease to the whole population of the aquaculture. The trajectory, speed, sudden changes in the orientation, etc., can be indicators of fish behavior. For example, sluggish or indolent fish movements may indicate that a fish is sick or simply not hungry while hyperactive fish that are afraid to reach food, may indicate that they are stressed, or feel threatened. Until recently, the measurement of fish dimensions, weight, etc., was performed manually and invasively, by taking sample fish out of the water. This is a manual, time-consuming, high-cost, inaccurate and harmful procedure for the fish. Fish tracking is almost impossible with the naked eye without infrastructure to capture and analyze underwater videos of adequate duration. Tasks like fish tracking and classification, morphological feature estimation and behavior monitoring are also important in observing the population of various species in rivers or the open sea. For example, the population of

a fish species can be estimated from the density of the detected fish in an image frame. A fish species variety can be identified from its shape and skin texture or color. Fish tracking can provide information about the behavior and potential stress of the fish in their natural environment. Estimating fish dimensions may also be useful in an open water environment for the assessment of a fish population condition. The morphological feature measurement is based on fish shape or contour alignment in order to measure the dimensions, detect malformations and locate parts of the fish that are of particular interest such as the eyes and the gills. Moreover, morphological feature measurement methods can also be exploited in fish product processing laboratories to estimate fish sizes, freshness, recognize species variations, etc.

A review of smart aquaculture systems is presented in [1] where several processes are described for breeding, nursery to grow out stages of cultured species, preparation of cultured water resource, management of water quality, feed preparation, counting, washing the cultured systems, etc. Another review of computer vision applications for aquaculture is presented in [2]. These applications include fish and egg counting, size measurement, mass estimation, gender detection, quality inspection, species and stock identification, monitoring of welfare and behavior. The trends in the application of imaging technologies for the inspection of fish products are examined in [3]. The reviewed image-processing approaches are classified by the position of the light source (reflectance, transillumination, transflectance, etc.). The applications examined include rigor mortis, tissue and skin color as well as morphology to determine fat, firmness, shape, wounds, blood detection, etc.

The approaches for morphological feature measurement that have been proposed in the literature, are often based on the contour of a fish or on the localization of specific landmarks. Both image processing and deep learning approaches have been proposed that can either achieve a high frame processing speed or a high accuracy. The applications that achieve high accuracy in the estimation of morphological features are often tested on datasets with high-quality images where the fish are clearly visible. In these datasets, fish are captured either in a controlled laboratory environment or underwater with expensive cameras and good environmental conditions (minimal reflections, calm sea, no murky waters, etc.). Moreover, the size of the fish in these datasets is reasonably large to discriminate the details of their bodies. The motivation for this work was to adapt a popular human face shape alignment Machine Learning (ML) technique for fish shape alignment with hardware acceleration support in order to achieve both high speed and accuracy. Additional software modules and applications have been developed that either support the proposed morphological feature extraction method (e.g., orientation classification, landmark editor) or offer additional services such as fish tracking. Moreover, we also developed a dataset with low-contrast underwater images and videos, displaying relatively small fish in murky waters, with intense reflections and refraction, limited visibility, turbulent waters, etc. Thus, testing our framework with this dataset provided experimental results under worst case conditions.

In the system described in this paper, a three-stage approach is followed to detect fish in low-quality image frames where fish cannot be easily discriminated from the background. The input image frame is segmented to extract the bounding boxes of the detected fish as separate image patches. In the second stage, each patch is classified to a draft orientation in order to select the corresponding pre-trained ML model that can align a number of landmarks on the fish body. In the third stage, the shape (and potential malformations) of the fish can be recognized from the located landmarks, in order to measure fish dimensions, to classify fish and to map fish body parts of special interest (eyes, gills, etc.).

The first stage of the proposed system is based on the open source, fish detection, deep learning method presented in [4]. Although detailed instructions are given in [4] on how to train a customized fish detection model, the available pre-trained one performed very well even with the low-resolution images of our dataset. Therefore, the pre-trained model was stored on the target platform and was called from an appropriate Python script that has been developed for image segmentation. The output of this script is a number

of image patches and each one of these patches contains a single fish. The coordinates of these patches in the original input frame are also extracted. Each one of the image patches is classified in a draft fish orientation category, following high-speed methods that are based on OpenCV [5] services. The coordinates of the patches can be used to track the movement of the fish in successive frames. The short history of fish positions that have been detected in the first stage can be used to estimate extended bounding boxes for the intermediate positions through interpolation in order to bypass the time-consuming fish detection process in some frames.

The extracted patches from each image frame are used as inputs to the last stage of the developed system that performs shape alignment. Aligning a number of landmarks is based on the ML approach called Ensemble of Regression Trees (ERT) presented by Kazemi and Sullivan in [6] and is exploited in popular image-processing libraries such as DLIB [7] and Deformable Shape Tracking (DEST) [8]. The DEST library was exploited in our previous work [9] for driver drowsiness applications. The source code of the DEST library was ported to Ubuntu and Xilinx Vitis environments to support hardware acceleration of the shape alignment process on embedded targets. In the context of the present work, the DEST library has also been ported to Microsoft® Visual Studio 2019 environment for fish shape alignment.

Previous work on fish morphological feature measurement has also been presented by one of the authors in [10] but it concerned different fish species and employed different approaches for image segmentation, pattern matching and contour recognition. The fish contour detection method followed in [10] was not based on shape alignment and ERT. Specifically, three methods were proposed in [10] for fish contour recognition: (a) Pattern Matching (PM), (b) Mask R-CNN applied on a Binary Mask Annotation (BMA) or on (c) a Segmented Color Image Annotation (SCIA). An absolute fish dimension estimation based on stereo vision was presented in [10] that is also applicable in the framework presented here. The approaches (BMA, SCIA) presented in [10] exhibited in most cases, a much higher error in the estimation of fish dimensions, than the present solution. All the alternative methods (PM, BMA, SCIA) presented in [10] required a much higher frame-processing latency (in the order of seconds) than the current approach thus, they could not be exploited for real-time applications.

In the present approach, the accelerated shape alignment method required for the morphological feature extraction showed a latency of a few ms on an FPGA platform or less than 0.5 μs on an Intel i5, 3 GHz processor. The error in landmark position estimation is in the order of 5% owed mainly to the low-contrast and quality of the images in our dataset. Advanced histogram equalization techniques for contrast enhancement can be found in [11]. Since no clear reference photographs are available in our case, contrast enhancement could be achieved with No Reference methods like the one presented in [12]. In [12], information maximization is attempted by removing predicted regions (sky, sea) and estimating the entropy of particular unpredicted areas via visual saliency. From a global perspective, the image histogram is compared with the uniformly distributed histogram of maximum information to find the quality score of the applied contrast mechanism. Contrast enhancement methods will be employed in our future work to reduce the landmark position estimation error. In the present work however, we use the developed dataset with low-quality images without any enhancement to test the efficiency of our shape alignment method under worst case conditions.

The contribution of the present work can be summarized as follows: (a) shape alignment based on ERT models is adapted to fish shapes, for high-precision morphological feature estimation, (b) ERT models with different parameters are trained to find a tradeoff between speed and accuracy, (c) a different ERT model can be trained for each fish orientation, (d) a fish detection method efficient for low-contrast images is employed and adapted for local execution in the proposed framework, (e) fish tracking is supported exploiting interpolation and orientation classification results, (f) hardware and software acceleration techniques are implemented for shape alignment and others are also applicable for fish

detection in order to support real-time video processing, (g) a new landmark editor has been developed to easily prepare the training set and ground truth data, and (h) a new public dataset with realistic photographs and videos has been developed.

This paper is organized as follows. The related work is presented in Section 2. The materials and methods used are described in Section 3. More specifically, the dataset, tools and target environment are presented in Section 3.1. The general architecture of the proposed system is described in Section 3.2. The employed fish detection and the fish orientation methods are described in Sections 3.3 and 3.4, respectively. The methodology for implementing fish tracking is described in Section 3.5. The ERT background and the customized DEST package used for fish shape alignment and morphological feature extraction are described in Sections 3.6 and 3.7, respectively. The experimental results are presented in Section 4. A discussion on the experimental results follows in Section 5 and the conclusions are presented in Section 6. All abbreviations used throughout this paper are defined in Abbreviations.

2. Related Work

Several approaches have been proposed concerning the estimation of fish freshness in a controlled laboratory environment, based either on sensors or image processing. In [13], various sensors that have been used in the literature for freshness estimation are reviewed. These sensors include biosensors, electric nose or tongue, colorimetric sensor array, dielectric and various sensor for spectroscopy (nuclear magnetic resonance, Raman, optical, near infrared, fluorescence, etc.). Quality management systems have also been proposed for freshness, safety, traceability of products, adopted processes, diseases and authenticity [14]. In [15], the freshness of Scomber japonicus (mackerel) stored at a low temperature is assessed from the correlations between the light reflection intensity of mackerel eyes and the volatile basic nitrogen content. The assessment of fish freshness from the color of the eyes is also examined in [16]. In this approach, a handheld Raspberry PI device is used to classify the freshness of a fish into three categories (extremely fresh, fresh, spoiled) based on pixel counting.

Fish classification and counting from underwater images and videos is another major category where several approaches have been proposed in the literature. In [17], fish appearing in underwater images are classified in 12 classes based on Fast Regional-Convolutional Neural Networks (Fast R-CNNs). Similarly, in [18], You-Only-Look-Once (YOLO) [19] and Gaussian Mixture Models (GMM) [20] are compared for the classification of 15 species with an accuracy between 40% and 100% (>80% in most cases). Lekunberri et al. [21], count and classify various tuna fish species transferred on conveyor belt with 70% accuracy. Their approach is based on various types of neural networks (Mask R-CNN [22], ResNet50V2 [23]) while the size of tuna fish, ranging from 23cm to 62cm, is also measured. Underwater fish recognition is performed in [24] with an accuracy of 98.64%. Similarly, fish recognition from low-resolution images is performed in [25] with 78% precision.

Morphological feature estimation is often based on active and passive 3D reconstruction techniques. The active techniques are more accurate but require expensive equipment such as Lidars, while passive techniques employ lower cost cameras. Challenges of passive 3D reconstruction include the accurate depth estimation from two images that have been retrieved concurrently, occlusions, patterns and saturate areas that may cause confusion. In [26], a system based on stereo camera is described for accurate fish length estimation and fish tracking. A monocular 3D fish reconstruction is presented in [27], where successive images are used from fish carried on a conveyor belt in order to measure their size. CNNs implemented on Graphical Processing Units (GPUs) are used for foreground segmentation and stereo matching. A median accuracy of less than 5mm can be achieved using an equivalent baseline of 62 mm.

In [28], Facebook's Detectron2 machine learning (ML) library has been employed for object detection and image preprocessing to generate 22 metadata properties including morphological features of the examined specimens with error rates as low as 1.1%. Otsu

threshold is used for segmentation of relatively simple images and pattern matching to locate the eye. If the fish is detected without an eye the images are up-scaled.

Fish tracking (and classification) can be performed with both optical and sonar imaging as described in [29]. Using sonar imaging is the only way to monitor fish at night time. In this approach, the Norfair [30] tracking algorithm in combination with YOLOv4 [31] are used to track and count fish. The employed sonar equipment is dual-frequency identification sonar (DIDSON) that exploits higher frequencies and more sub-beams than common hydroacoustic tools. The use of DIDSON has also been described in [32] for the detection of fish morphology and swimming behavior. In this approach, fish must be large enough and within an adequate distance thus, it is not appropriate for counting small fish. Fish length should preferably be around 68 cm, otherwise an estimation error ranging from 2–8% was measured for different size fish (40–90 cm). In [33], optical and sonar images are also employed for fish monitoring.

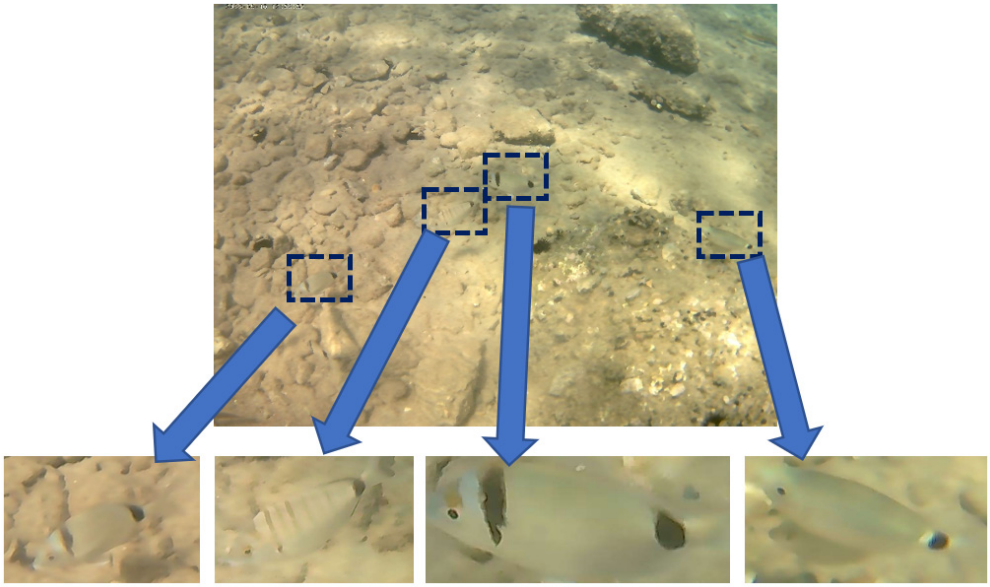
3. Materials and Methods

3.1. Dataset, Tools and Target Environment

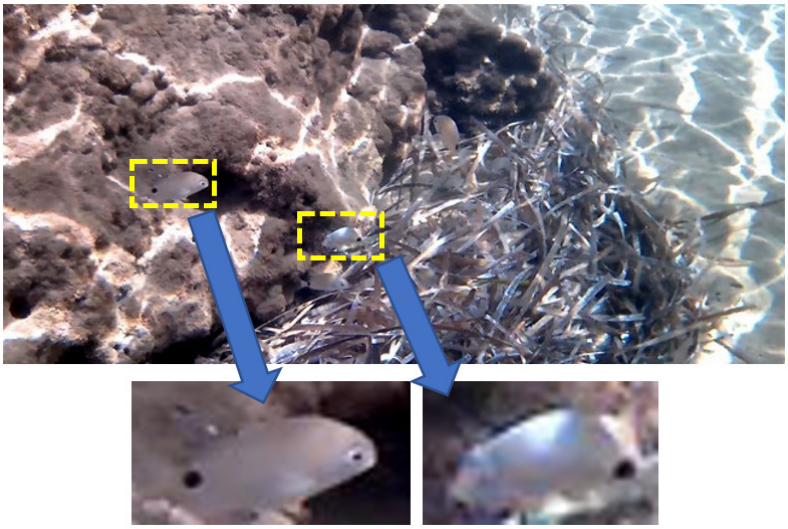
The image datasets used in similar fish monitoring applications usually consist either of images that have been captured in a laboratory environment under controlled light exposure, or high-resolution underwater images where the fish are clearly visible. The images and videos used in the context of this paper have been retrieved using a low-cost camera (TurboX Act201), their image resolution is 4608 $\times$ 3456 and the resolution of the video frames is 3840 $\times$ 2160, with a frame rate equal to 30 frames per second (fps). The photographs and videos of the developed dataset have been captured in two regions in Greece (Lampiri, Achaia and Gavathas, Lesvos island) and display Mediterranean fish species mostly seabream variations such as diplodous sargus annularis (white seabream), diplodus annularis seabream (spawn), oblada melanura (saddled seabream), pagrus pagrus (common seabream), etc. The dataset we developed is called Underwater Videos and Images with Mediterranean Fish Species in Shallow Waters (UVIMEF) and its first version is made publicly available in [34]. An example photograph and a video frame with the corresponding fish patches extracted, are shown in Figure 1.

The trained ERT models align L = 18 landmarks on the fish body and can also be used with other species that have similar shape and color such as sithognathus mormyrus, sparus aurata, dicentrarchus labrax, etc. New ERT models can be easily retrained for different number of landmarks and fish shapes. The images/videos of the developed dataset have been captured both during sunny and cloudy days with calm or stormy seas at a depth between 0.5 and 3 m.

From the 60 videos and 720 photographs of UVIMEF, 400 image frames have been used to generate 322 single fish image patches. The training of our ERT models has been performed using 270 of these image patches and the rest of them have been used for testing. The fish species appearing in these image patches are listed in Table 1. The number of images per species is not balanced because all of the fish species are similar in shape as far as the position of the landmarks is concerned. Thus, the training of the ERT model is not significantly affected by the number of fish per species listed in Table 1. The dataset should be balanced however, if fish species with radically different body shape had to be supported. The last row of Table 1 lists fish that do belong to the categories listed above but still have similar body shape such as Sarpa salpa, or Dentex dentex or fish that cannot be recognized due to extremely low contrast. These species are not listed in separate rows because their population is too small. The fish in these patches have a specific orientation e.g., horizontal body-facing left, or with tilt facing up-right, etc. As can be seen from Figure 1, the fish in these image frames are hardly distinguishable from their background. Consequently, the fish detection, monitoring and morphological feature extraction methods presented in this paper are applied in worst case conditions.



(**a**)



(**b**)

Figure 1. Fish patches extracted from a 4608 × 3456 resolution photograph (**a**) and from a 3840 × 2160 resolution video frame (**b**).

Table 1. Fish species in the dataset used.

Fish Species	Number of Image Patches
Diplodus Sargus	105
Lithognathus mormyrus	61
Diplodus annularis	62
Oblada Melanura	60
Other	34

The host computer used in the experiments is an ordinary Dell laptop with Intel(R) Core(TM) i5-1035G1 CPU @ 1.00 GHz and 16GB RAM. The artificial intelligence (AI) inference for fish detection, the orientation classification script, the landmark annotation editor (LAE) and the training/testing of the ERT model for shape alignment were installed on this computer. As will be discussed in Section 3.7, the target board can be an embedded platform such as a Xilinx ZCU102 with ZynqMP Ultrascale+ field programmable gate array (FPGA) in order to support hardware acceleration. Concerning the programming languages

used for the development of the system presented in this paper, the fish detection and orientation scripts are implemented in Python while the shape alignment is implemented in C++ (Visual Studio 2019 environment). The new landmark annotation editor (LAE) has also been developed in Visual Studio 2019, in C# as a Universal Windows Program (UWP) and is presented in Section 3.7.

3.2. General Architecture of the Proposed System

The architecture of the proposed fish monitoring and morphological feature estimation system is shown in Figure 2. The input of the system is either a single photograph or a frame from an input video stream. A customized Python script based on the open source code and the pre-trained fish detection model presented in [4], analyzes the input image frame and detects the bounding boxes of the fish. The coordinates of these bounding boxes are used to crop patches displaying a single fish. These patches are used as input to the next stage that detects fish orientation. The orientation of the fish is necessary in order to select the appropriate pre-trained ERT model for shape alignment. The ERT ML method employed for shape alignment is based on the correction of a mean shape stored in the pre-trained model. This correction procedure cannot support a rotation in the mean shape by more than $\pm 20^{\circ}$. For this reason, different ERT models are trained for each direction and the appropriate model is selected based on the orientation classification results for the shape alignment process. The output of shape alignment stage is the set of 18 landmark coordinates that determine the location of fish body parts of particular interest such as the mouth, the eyes, the caudal fin, etc. From the distance of these landmarks, fish morphological features such as the relative fish length/height and its shape can be extracted. The shape alignment process has been accelerated both in hardware and software level, based on the principles described in [9].

The AI inference of the model used for fish detection can also be accelerated in hardware since it is a time-consuming process. However, an alternative way based on interpolation can be followed to accelerate the fish detection process without special hardware support. Its aim is to define extended bounding boxes in frames where fish detection is not applied to save time. If the shape alignment performed in one of the interpolated bounding boxes fails then, the results based on the extended bounding box approximation are invalidated. A small image frame buffering (e.g., of 10 image frames) is necessary to guarantee that all the processing results are correct. The successive positions of a fish are registered to track its movement in order to study its behavior.

3.3. Employed Fish Detection Approach

The fish detection method employed in this paper is based on the tools presented in [4]. With this repository, the developer has the option either to use a pre-trained model or customize a new model, following the instructions given. The training process can be performed in an Ubuntu environment using a docker and consists in short of the following steps: (a) a tensor flow object detection GitHub repository is cloned, (b) the training dataset has to be prepared in *tfrecord* format, (c) a new label map should be created, (d) an existing model like Faster RCNN has to be downloaded, (e) a copy of an appropriate configuration has to be created and customized, (f) the model should be trained using an available Python script, (g) the validation results can be observed, (h) a graph in pb format has to be extracted and used as inference, (i) the inference can be called within Jupyter notebook or Collab environment. The creator of the repository in [4] also offers a pre-trained *pb* model as the one that can be generated in step (h). This model proved extremely sensitive for our dataset and even fish that were hardly recognized with the naked eye have been detected by this model.

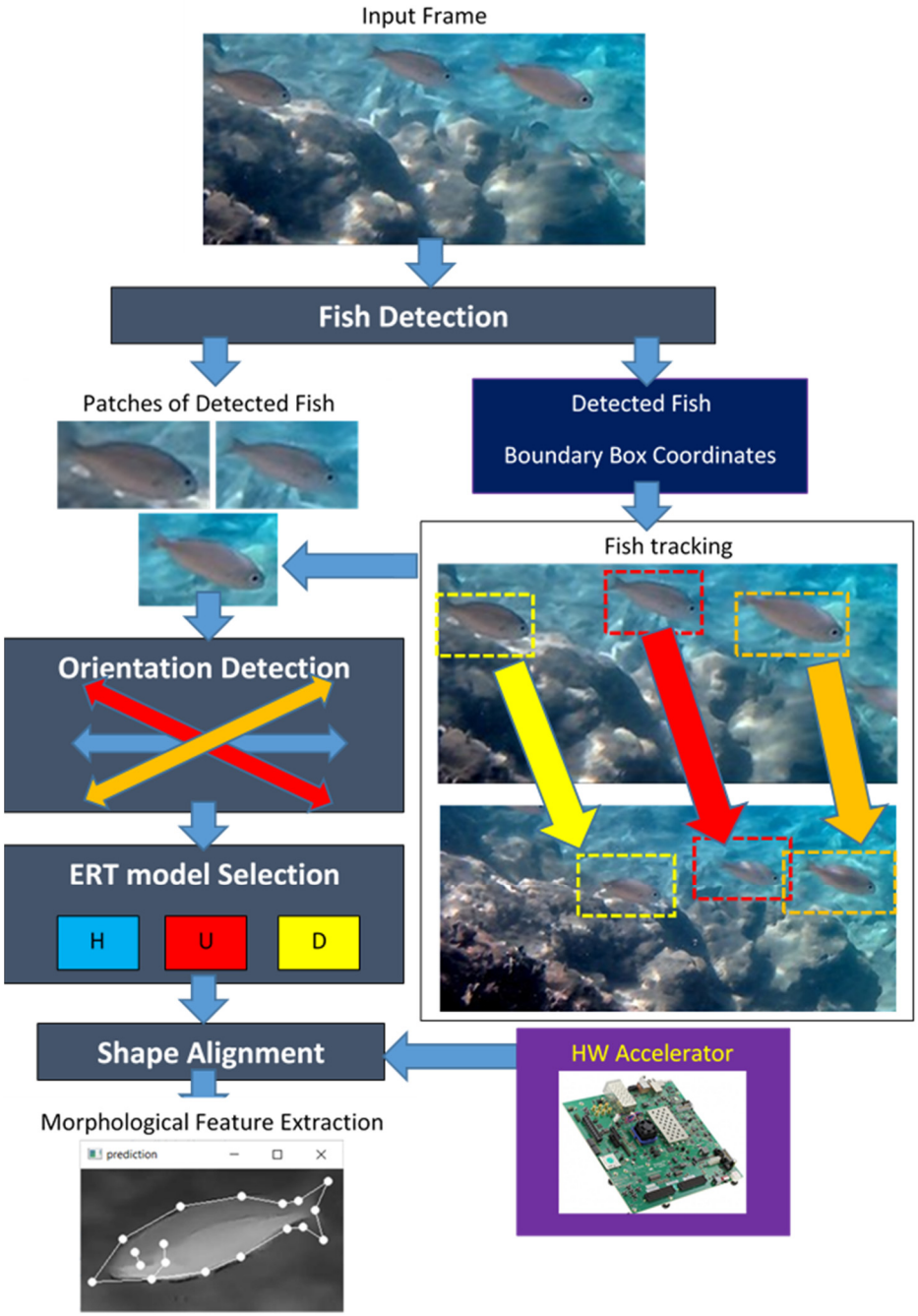


Figure 2. The architecture of the developed system for fish monitoring and morphological feature extraction.

Certain Python scripts offered in [4], have to be executed when the fishes in a new image frame have to be recognized. The modification of these Python scripts was necessary in order to run the inference locally, i.e., outside a Jupyter environment and integrate the fish detection process in our flow. We extract the coordinates of the bounding box of a detected fish, in order to crop the initial image and generate a patch that will be used in the orientation detection and the shape alignment stages of the developed system. The bounding box coordinates are also passed to the fish tracking stage. Figure 1a shows the detection of three fish from an input photograph and Figure 1b, the detection of two fish from a video frame.

The drawback of this fish detection approach is the high latency required for the analysis of a single image. After loading the inference model, the Python script that performs the inference needs about 1.6 s to analyze each video frame, on an Intel i5, 1 GHz processor and about L_{fd} = 1 s on an Intel Core i5-9500 CPU @3.00 GHz, 6 core processor with 16 GB RAM. Video processing in real time would not be possible if fish detection had to be applied to all the input frames. One approach would be to accelerate the inference on a GPU or an embedded hardware platform. Lightweight or pruned NN models can also be trained following the procedure described in [4] to achieve higher speed with a slightly lower accuracy. Another lower cost approach is to buffer the video frames, apply fish detection in regular time intervals and interpolate the position of the fish between successive fish detections. This method will be examined in more detail and will be called henceforth: Fish Position Interpolation (FPI).

In FPI, it will be assumed that fish detection takes place every T_{fd} seconds. In modern multi-core processors T_{fd} can be as low as L_{fd}, since one thread can run the inference and another parallel thread can run the FPI between successive fish detections that took place with a time distance of T_{fd}. The fish orientation between successive fish detections does not necessarily have to remain the same, since the orientation detection may also be applied to the interpolated bounding boxes. However, if fish is found to have the same draft orientation between successive fish detections, the orientation detection procedure does not need to be repeated in the interpolated positions to speed up the whole process. For example, as can be seen in Figure 3, a fish is moving in the same direction (denoted by the orange arrow) for 2 s.

Figure 3. Fish movement in video frames with time distance of 2 s.

The FPI procedure is described in more detail in Figure 4. If the input stream has a standard frame-processing speed of 24 fps, it is assumed that shape alignment is adequate to be applied every six frames, i.e., every 0.25 s. More intermediate shape alignments may be applied in shorter intervals but they are not expected to give more information about the track, the behavior and the morphological features of a fish. Thus, if a fish detection takes place every T_{fd} = 1 s, at frame No. 0 (F0) then, the bounding boxes of the fish in frames No. 5 (F5), 11 (F11) and 17 (F17) have to be interpolated. For this purpose, the center of the 1st bounding box (F0) and the center of the bounding box generated from the next fish detection are located. The straight line connecting these bounding box centers is split according to the number of interpolations I_f that will be performed. As shown in Figure 4, the distance between two successive fish detections is split into four equal segments in order to define the center of the I_f interpolated frames (F5, F11 and F17). The size of the interpolated bounding boxes is incremented by a fraction z. The value of z was experimentally selected between 10% and 20%. Obviously, if z = 20% is selected, the dimensions of F17 will be 172.8% larger than the original bounding box F0. If the size of the bounding box becomes too large compared to the fish dimensions displayed in this box, the shape alignment accuracy will be degraded. On the other hand, if z = 10%, F17 dimensions

will be only 133.1% larger than the F0 bounding box. The shape alignment accuracy in this case would be higher but there is also a higher risk of obtaining an interpolated bounding box where parts of the fish will be excluded, or even a bounding box that does not display a fish at all.

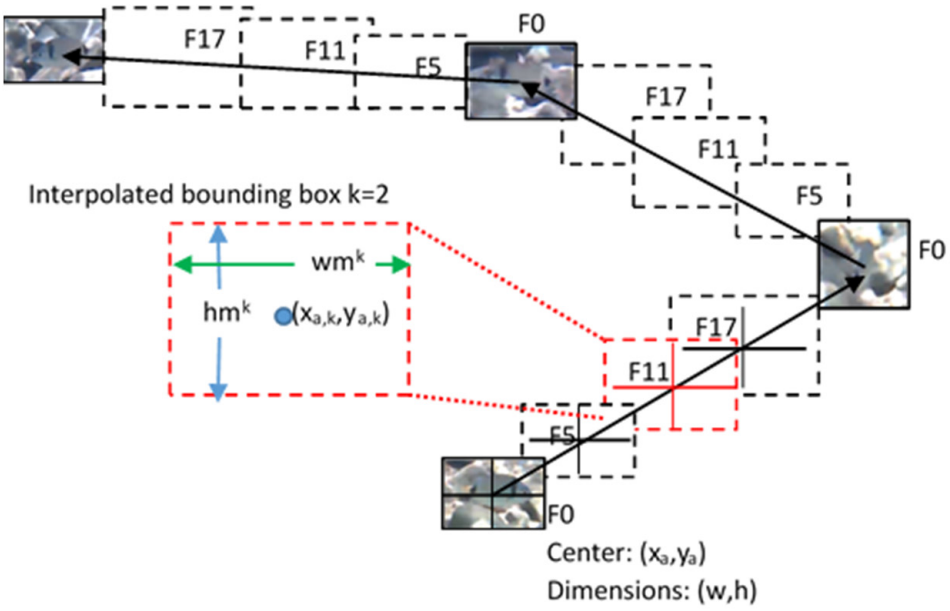


Figure 4. Fish Detection Interpolation (FPI).

The general formal definition of the interpolation between two successive fish detection bounding boxes with centers (x_a, y_a) and (x_b, y_b) follows. If I_f interpolation bounding boxes are defined, the center $(x_{a,k}, y_{a,k})$ of the k-th $(0 < k \leq I_f)$ interpolation bounding box is defined as:

$$(x_{a,j}, y_{a,j}) = (x_a + k\frac{x_b - x_a}{I_f + 1}, y_a + k\frac{y_b - y_a}{I_f + 1}) \quad (1)$$

If the width and the height of the original bounding box is w_{bb} and h_{bb}, respectively, then, the corresponding dimensions $w_{bb,k}$ and $h_{bb,k}$ are defined as:

$$(w_{bb,k}, h_{bb,k}) = (w_{bb} \cdot z^k, h_{bb} \cdot z^k) \quad (2)$$

3.4. Orientation Classification Method

The ERT model [6] that will be used for the shape alignment and consequently for the measurement of the morphological features of the fish has to be trained on fish that have similar orientation. The human face shape alignment approach presented in [6] and widely used in popular libraries like DLIB [7], defines the facial shape with 68 landmarks and is more tolerant on the tilt of the head due to the symmetry of the shape. The face shape is symmetrical since every landmark has a horizontal mirror landmark. This fact is actually exploited when training an ERT model, since the dataset is augmented with the mirror images. The 2D shape as determined by the side view of the fish is the most important aspect, since special Regions of Interest (ROIs) can also be located on the body, besides the estimation of fish length and height. Such ROIs are the eyes and the gills since the health of the fish can be assessed from the color of these regions. Moreover, fish shape malformations can be detected from the side view. The employed ERT ML method for shape alignment is tolerant to a tilt in the shape of about 20°.

An option would be to detect the 1D orientation of the fish on the image plane and rotate or mirror it before performing shape alignment. Horizontal mirroring is actually an applicable operation that does not affect the achieved accuracy, i.e., a model trained with fish facing left are expected to have the same accuracy as a model trained with fish facing right. However, in the general case, a fish with tilt higher than 20°, may not have exactly the same shape as a fish captured horizontally, as shown in Figure 5 (depending also on the camera angle of view). For this reason, different ERT models may have to be trained for a small number of distinct orientations in order to achieve an accurate shape alignment. Even if we are interested only in images with fish in horizontal position, it is necessary to detect the draft orientation in order to reject images where the fish is not horizontal or use them merely for fish tracking and not morphological feature extraction.

Concerning the approaches presented in the literature for object orientation, 1D orientation of vehicles is achieved in [35] based on a YOLO network architecture with a modified output fully connected layer (enriched with additional orientation parameters). Deep CNNs (DCNNs) are used in [36] for a continuous object orientation estimation between 0 and 360 degrees. Three continuous orientation prediction approaches based on DCNNs are described in [36]. In the first two, the orientation is represented as a point on a unit circle and either L2 loss or angular difference loss is minimized. In the third more efficient method, the continuous orientation estimation task is transformed into a set of discrete orientation estimations (as also required in our case) and then, the discrete orientation is converted to a continuous one with a mean-shift algorithm.

Principal Component Analysis (PCA) has also been proposed for feature extraction of a dataset. PCA accepts as input all the features of a dataset, but only considers a subset of features that are important to detect correlations between the selected features, resulting in a reduced dimensionality of the dataset [37]. The eigenvectors presented in the PCA process determine the directions along which, the data have the highest variance [38]. In [39], OpenCV services are used to determine the direction of objects in images that are well discriminated.

A brief introduction to the PCA process follows. Let us suppose we have a matrix $X_{rxn} = [X_1\ X_2\ \ldots\ X_n]$ with measurements. X_i is a sub-vector of r elements s_j, which is necessary for the calculation of principal components. The covariance matrix with each X_i needs to be calculated for the measurement of the variability or spread in the dataset.

$$(R)_{r\times r} = \sum_{i=1}^{r} X_i X_i^T \tag{3}$$

The singular value decomposition (SVD) of $(R)_{rxr}$, is defined as follows:

$$(R)_{r\times r} = US_dV^T \tag{4}$$

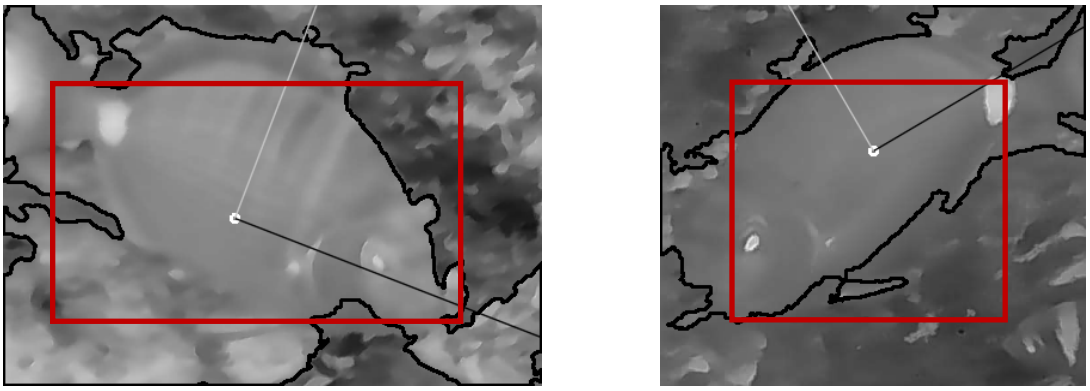
S_d is a diagonal matrix with the singular values of R on its diagonal, while V is a matrix having the right singular vectors of R as columns. $U = [U_1\ U_2\ \ldots\ U_n]$ is a feature vector (matrix of vectors). The columns of U are the left singular vectors of R.

The first m sub-vectors from the vector U are preserved while the rest are discarded. In other words, the first m eigenvectors (ordered by magnitude) are used in order to create $(U_{reduced})_{nxm} = [U_1\ U_2\ \ldots\ U_m]$, $m < n$. Those elements correspond to the principal components that can be used as a compressed version of the original input. More specifically, the transpose of the vector $(U_{reduced})_{nxm}$ can be multiplied with the original data set:

$$(Y_i)_{m\times 1} = (U_{reduced}^T)_{m\times n}(X_i)_{n\times 1} \tag{5}$$

The original data can be restored with the following equation:

$$(X_i^{received})_{n\times 1} = (U_{reduced})_{n\times m}(Y_i)_{m\times 1} \tag{6}$$



(**a**) (**b**)

Figure 5. The Segmentation of the image using OpenCV Otsu threshold and applying PCA with m = 2 to obtain the angles of the eigenvectors with the higher magnitude (black and white line). In both images the tilt denoted by the black line is recognized successfully but the direction in (**a**) is correct while in (**b**) is incorrect.

The detection of the draft fish orientation in our approach has to be performed with high speed and thus, DL methods are not appropriate due to their high latency. OpenCV PCA and image segmentation methods can offer an alternative faster solution with good accuracy. More specifically, the PCA method described in [39] has been adapted in our framework. In 95% of the cases the tilt of the fish is detected accurately in the patches extracted by the fish detection stage. However, only in 45% of the cases, the detected direction that the fish is facing at, was correct. The employed PCA method, converts the input image patch in grayscale and resizes it. All resized patches are 320 pixels in height.

The image patch is inverted if the fish color is darker than its background. In order to detect the brightness of the fish, the average gray level of the internal part of the image is compared to that of the image border zone, separated by the red squares in Figure 5. The border zone is defined with a thickness of $B = 20$ pixels. Having in mind that the image patches are actually bounding boxes of the detected fish, it is expected that most of the fish body is mainly displayed in the internal region while the background in the border zone. If the average gray color of the internal region (fish) is darker than that of the border zone (background) then, the image color is inverted as is the case in both images of Figure 5. The image is then segmented using Otsu's threshold [40] and the contour surrounding the largest area is used as input to the OpenCV PCA method.

The output is an accurate estimation of the fish tilt axis but the direction is not detected accurately. Therefore, additional segmentation methods are deployed to determine the direction the fish is facing. Having in mind that the ERT model can detect shapes with a $\pm 20^{\circ}$ tilt, it is sufficient to classify fish captured from the side view in 4 different orientation categories: fish facing up-left (Q0), up-right (Q1), down-left (Q2) and down-right (Q3). We assume that fish cannot be captured in a 90° vertical tilt since it is not natural for a fish to swim in this direction. Using the anatomy of the fish it is expected that if most of the fish body is found in the left half of the image patch, the fish is facing left (Q0 or Q2), otherwise it is facing right (Q1 or Q3). This simple method is called hereafter Coarse Orientation Detection (COD). It is implemented by comparing the area of the fish body in the left and right areas of the image, i.e., Q0 + Q2 is compared to Q1 + Q3. Symbols Q0, Q1, Q2 and Q3 are used to represent the number of pixels mapped to fish body in the corresponding quadrants. Extending this principle to the individual Q0, Q1, Q2, Q3 quadrants, a Fine Orientation Detection (FOD) takes place. If for example, the COD decides the fish is facing left then the fish is facing up-left if Q0 > Q2 (i.e., the area of the fish in Q0 is higher than the area in Q2), otherwise it is facing down-left. The COD and FOD processes are demonstrated in Figure 6.

(**a**) (**b**)

Figure 6. COD and FOD methods. (**a**) Q0 + Q2 > Q1 + Q3 (COD: fish facing left), Q0 > Q2 (FOD: fish facing Up-Left. (**b**) Q1 + Q3 > Q0 + Q2 (COD: fish facing right), Q3 > Q1 (FOD: fish facing Down-Right).

A fourth orientation detection method employed is based on template matching. More specifically, a template of the fish eye is used to locate the position of the eye and consequently the orientation of the fish. Based on the resizing of the image patches described earlier and the fact that the fish have similar dimensions after resizing, a 30 × 30 pixel template of the eye is used. The OpenCV template matching procedure is repeated for resized eye templates of 28 × 28 and 32 × 32 pixels. The direction shown by the majority of the three template matching procedures is used as output of this orientation detection method. Figure 7 shows examples of successful and unsuccessful eye template matches. In some low-contrast images the fish eye is not visible at all, while the eye template can easily match background objects or even other body parts such as the caudal fin of the diplodus annularis as shown in Figure 7d.

(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 7. Fish eye template (**a**), successful matching in (**b**,**c**), caudal fin confused with fish eye in (**d**) and background object confused with fish eye in (**e**).

To improve the orientation classification accuracy of the 4 orientation detection methods described above, combinations of these methods can be examined as will be described in Section 4.

3.5. Fish Tracking

The fish tracking supported in the framework of this work registers the coordinates of the bounding boxes and the orientations of the fish found in successive frames. The target is to determine the track of an individual fish and avoid confusion with tracks of other fish swimming in the neighborhood. To understand the concept of the adopted approach, the successive frames shown in Figure 8 can be considered. These frames have a time distance of 1 sec which is rather large and are captured from a camera that was not stable as required for the case of fish track monitoring. However, they contain a number of fish all facing towards the same direction (bottom-right) making the explanation of the tracking algorithm easier. Let c_i^t be the coordinate pairs of the top left corner of the bounding box i in frame t. It is assumed that the bounding box j corresponding to the same fish in the next frame $t + 1$ is the one that satisfies Equation (7), provided that the frames t and $t + 1$ have a relatively short time distance, e.g., less than 1 s.

$$i = argmin \| c_j^{t+1} - c_i^t \|_2 \tag{7}$$

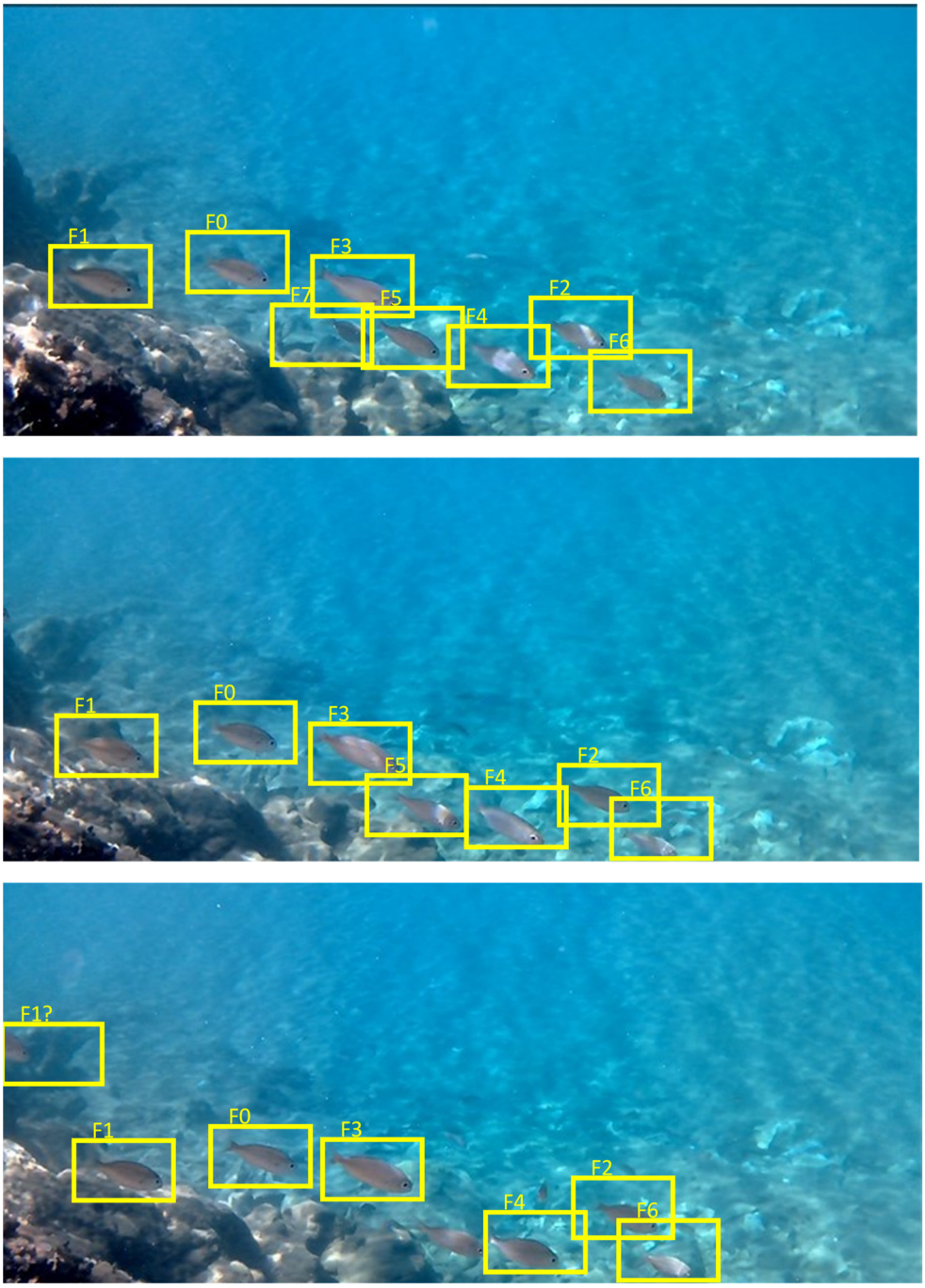


Figure 8. Monitoring fish in 3 successive frames. The recognized fishes are named F0–F7.

If the time distance between these two frames is too long, the correspondence of bounding boxes with fish in the next frame may not be accurate since: (a) the reference fish may have moved away and another one is now closer to the previous position of the reference fish and (b) the reference fish may have changed direction, or orientation, etc. In the top frame of Figure 8, seven fish are detected (F0–F8). The new position of these fish is monitored in the next two frames of Figure 8.

The fish names in the middle and bottom frames of Figure 8 were assigned based on the Euclidean distances estimated in Table 2. In each frame, the coordinates of the top-left bounding box corners are listed for the fish detected in these frames. In the first frame, the bounding boxes are assigned to names F0 to F7. In Frame No 2, the top-left coordinates of each bounding box are used in Equation (7) to estimate its Euclidean distance from each one

of the bounding boxes found in Frame 1. The shorter distance is used to assign fish names in the second frame as shown in the 4th column of Table 2. Each fish name is followed by its distance (in pixels) from the bounding box of the fish that it has been assigned to. Fish F7 was not recognized in Frame 2. Similarly, in Frame 3 the new bounding boxes are associated to the fish detected in Frame 2 using the shorter distances.

Table 2. Association of the coordinates of the bounding boxes detected in Frames 2 and 3 with the fish detected in Frame 1 of Figure 8.

Fish Frame 1	Frame 1 Y	Frame 1 X	Fish Frame 2	Frame 2 Y	Frame 2 X	Fish Frame 3	Frame 3 Y	Frame 3 X
F0	186	164	F0 (12)	197	165	F3 (12)	206	250
F1	190	63	F3 (52)	207	238	F0 (11)	197	176
F2	231	409	F1 (14)	203	69	F1 (8)	211	71
F3	190	242	F2 (14)	242	418	F2 (23)	243	441
F4	246	353	F4 (13)	259	355	F6 (10)	276	474
F5	230	285	F6 (12)	280	464	F1 (104)	117	10
F6	269	459	F5 (21)	249	295	F4 (10)	265	364
F7	231	254						

In Frame 3, fish F5 has not been recognized while a new fish appeared at the left side of the image and was associated to F1 since this fish is closer. For this reason, there are two bounding boxes associated with F1 and the system has to decide which one actually corresponds to F1. The fish orientation is used to solve this issue. Since all the fish in the three frames are heading bottom-right the new fish that appeared from the left is not associated with the fish that appeared in Frames 1 and 2 and is assumed to be a new fish. The exploitation of the orientation features in fish tracking is triggered when a bounding box in the current frame is assigned to more than one fish detected in the next frame. Among the bounding boxes assigned to the same fish name, the one with the smaller distance and in the right direction is selected. Of course, if there is no ambiguity, the fish can change orientation in the current frame and its association with a fish in the previous frame is only determined by the Euclidean distance. On the other hand, the confirmed direction that the fish is following in the last frames, can also be exploited in the orientation classification and consequently in the morphological feature extraction stage to obtain more accurate results.

More features can also be taken into consideration for accurate track monitoring: (a) the bounding boxes associated with fish from the previous frame should belong to fish of the same species, (b) the size of the associated fish in successive frames should match, (c) the position of the new bounding box should agree with the speed of the fish measured from previous frames. The study of how these features can be incorporated in our fish tracking procedure as well as its evaluation is part of our future work.

3.6. ERT Background

The face alignment method presented in [6] is based on an ML method called Ensemble of Regression Trees (ERT) and was originally used to align landmarks on human faces. We use this method to find the shape of a fish based on a set of $L = 18$ landmarks. When ERT is applied to a new image, the mean position of the landmarks stored in the trained model is gradually corrected in T_c cascade stages. In each cascade stage, N_t regression trees are visited. Each binary regression tree has $2^{Td} - 1$ nodes and in each node the gray level of a pair of pixels from a sparse representation of the input frame is compared. The left or right node of the regression tree that has to be visited in the next level is selected according to the comparison results. A correction factor is found in the leaves of each regression tree and this factor is used to correct the current landmark positions.

Let S be the shape of the fish i.e., the set of the $L = 18$ pairs of Cartesian coordinates, and $\hat{S}^t$, the estimated shape at the cascade stage t ($0 \leq t < T_c$). In the next cascade stage $t + 1$, the shape estimation $\hat{S}^{t+1}$ is updated from $\hat{S}^t$ with a correction factor r_t:

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t \tag{8}$$

The correction factor r_t is determined by the residuals between the current estimation and the trained shapes as well as the correction factors retrieved from the regression tree leaves. At the beginning of each cascade stage, the sparse image represented by N_r reference pixels is warped to match the shape of the fish within a process called Similarity Transform (ST). If q is a reference pixel and its neighboring landmark has index k_q, their distance δx_q is represented as:

$$\delta x_q = \|q - x_{k_q}\|_2 \tag{9}$$

The pixel q' in the sparse image representation that corresponds to q in the mean shape, is estimated as:

$$q' = x_{i,k_q} + \frac{1}{s_i} R_i^T \delta x_q \tag{10}$$

The parameters s_i and R_i are scale and rotation factors, respectively, employed in the ST process to warp the fish shape. For more information, please see [6].

3.7. Shape Alignment for Fish Morphological Feature Extraction

The ERT method described in the previous paragraph and its implementation in DEST [8] has been employed and adapted for fish shape alignment based on the L = 18 landmarks as shown in Figure 9. Landmark No. 1 marks the position of the fish mouth, landmarks 2–5, the upper part of the fish contour, landmarks 6–10, the perimeter of the caudal fin, landmarks 11–14, the lower part of the contour, landmarks 14–16, the position of the gill and finally landmarks 17–18, the position of the eye. These landmarks allow the estimation of relative fish dimensions and the location of regions of interest (ROIs) of the fish body, that are critical for the assessment of the fish health such as the eyes or the gills. The relative length of the fish can be determined as the distance (expressed in pixels) between landmarks No 1 and 8, while the relative height as the distance between landmarks No. 3 and 13. Absolute dimensions can be estimated if stereo vision is employed with a pair of cameras instead of a single one. More details on how this can be achieved may be found in our previous work [10]. The fish shape malformations can also reveal information about the health of the fish, the specific species variation of the fish, its growing conditions, etc. The malformations can be indicated by the position of certain landmarks like those in the upper and lower part of the fish contour. A denser shape representation with more landmarks may be necessary to detect malformations or variations in the species. The developed framework can be trained to align fish shapes with any number of landmarks without any modification. The process that has to be followed is depicted in Figure 10.

The training on the target fish species requires a number of images, each one displaying a single fish. These images can be the patches generated by the fish detection phase in Figure 2, from underwater images. Each one of these single fish images should be accompanied by an annotation text file in order to perform the training procedure. A new Landmark Annotation Editor (LAE) has been developed that can be easily used to create or edit annotations of any number of landmarks. The format of the annotations created by the LAE editor is compatible with the one required by the suite of the DEST tools [8]. The main window of the LAE tool is shown in Figure 9. An image can be loaded in this editor and a new annotation can start or a stored one can be edited. Landmarks can be easily moved with a single mouse click. The annotation can be stored in one of the supported formats. The LAE tool offers a number of data augmentation functions (image crop and mirroring) that can be useful in the development of new training datasets. The dataset consisting of {fish image, annotation files} pairs are used for the training of an ERT model using the

dest_train application of the DEST suite of tools. Several parameters (T_c, N_t, N_r, tree depth, learning factors, etc.) of the ERT model can be defined in the dest_train application. In this way, various ERT models can be trained on the same dataset, with different tradeoffs between speed and accuracy as the ones that will be defined in Section 4.

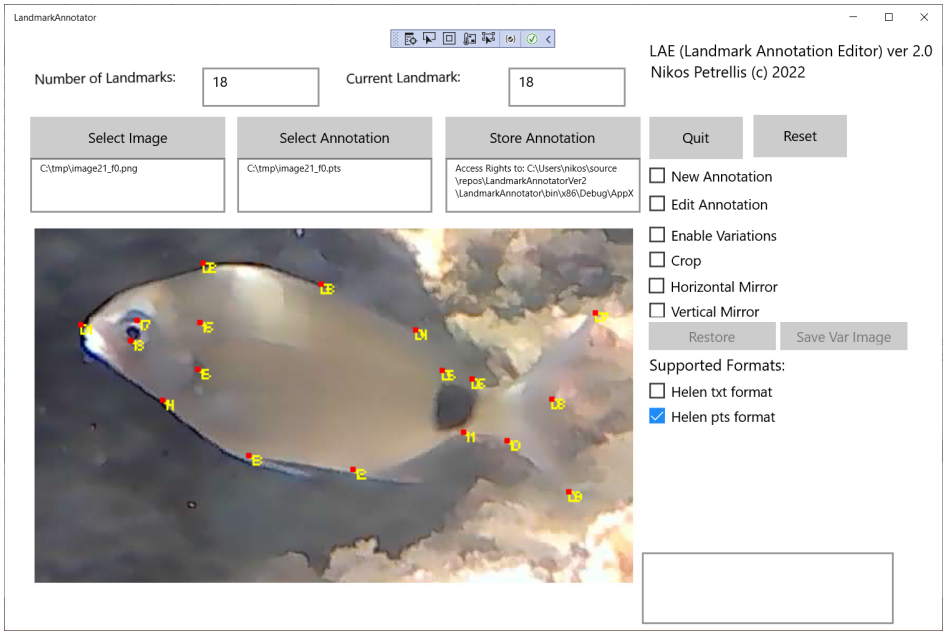


Figure 9. Fish shape alignment based on L = 18 landmarks and the developed LAE landmark annotator.

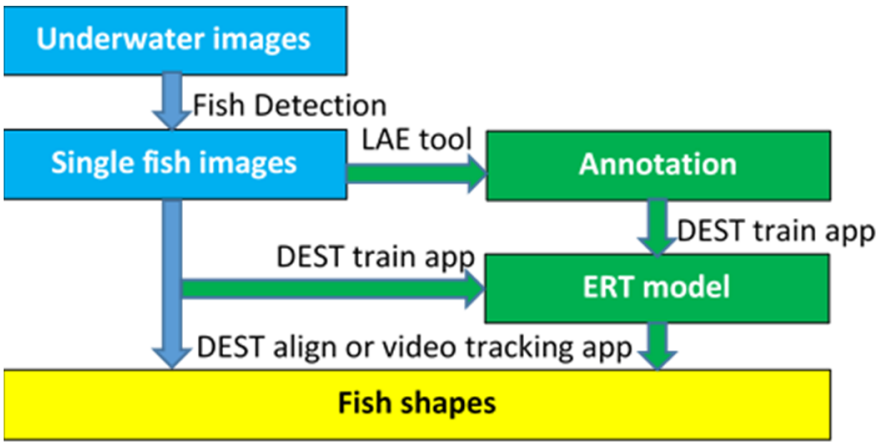


Figure 10. Fish shape alignment and training procedure.

The original DEST suite of applications [8] is written in C++ and has been ported in our previous work [9], in Ubuntu environment. The DEST application for video tracking has then been ported to a target Xilinx Vitis environment for hardware acceleration support in order to achieve a higher frame-processing speed. Although the porting to the Ubuntu and Xilinx Vitis environment has been performed for human face alignment in the context of a driver drowsiness application, the tools already developed were adapted with minor modifications for the alignment of fish shapes. For more convenience, the DEST suite of applications has also been ported to Windows environment using Microsoft Visual Studio 2019.

The resulting ERT model that has been trained, can be used with other DEST applications like the one that aligns shapes in single photographs (dest_align) or the one that aligns shapes in video frames (dest_video_tracking). In the process of porting the DEST applications' source code to both Ubuntu and Windows environments, a software

restructuring took place that allowed an acceleration in the frame-processing speed by more than 240 times: a 1920 × 1080 pixel frame processing latency was reduced from 116 ms to 475 µs on an Intel 6-Core i5-9500 CPU running at 3.00 GHz [8]. This significant acceleration in software was achieved by replacing time-consuming calls to the Eigen math library [41] with optimized C code. The original DEST source code called Eigen template library functions for matrix/vector operations, Jacobi rotations, Singular Value Decomposition (SVD), etc. Using Eigen library and well-defined C++ classes, ensures the integrity of the data values in any application and allows the operations to be described in a compact and portable way. However, excessive integrity checks and data type conversions were responsible for a high-latency overhead. Moreover, the Eigen and C++ classes and data types used in DEST were not appropriate for hardware synthesis and implementation on reconfigurable hardware using state-of-the-art tools such as Xilinx Vitis.

If even higher frame-processing speed is needed, the shape alignment latency can be further reduced by more than 50% if it is implemented on an embedded platform like a Xilinx ZCU102 development board with ZynqMP Ultrascale+ FPGA. Porting an ML method like ERT in hardware is not a typical hardware acceleration problem since a number of large arguments have to be passed to the kernel. Moreover, there are not many repetitive operations to be performed on these data. For this reason, the acceleration techniques have been focused on the reduction of the latency of bulk data transfers from the software (ARM processor) to the hardware kernel (FPGA).

In the present work, the tasks of the landmark prediction process that were implemented in reconfigurable hardware concern the traversing of the N_t regression trees of a cascade stage. Each one of these regression trees has depth T_d and $2^{T_d} - 1$ nodes. Therefore, the hardware kernel requires five arguments: (a) the indices of the pair of reference pixels that will be compared in each tree node (arguments: split1, split2), (b) the thresholds T_h in the difference between the gray level of the reference pixels that are compared at each tree node, in order to decide the next (left or right) node will be visited, (c) the correction factors r_t that will be added to the 18 pairs of landmark coordinates (only the correction factors at the 2^{T_d-1} regression tree leaves are needed) and (d) the N_r coordinate pairs of the reference pixels. The sizes of these arguments are listed in Table 3.

Table 3. Size of the arguments passed in the hardware kernel.

Argument	Number of Elements	Type
Split1/2	$2N_t\left(2^{T_d}-1\right)$	Short integer
Threshold T_h	$N_t\left(2^{T_d}-1\right)$	Floating point
Corr. Factors r_t	$2LN_t2^{(T_d-1)}$	Floating point
Reference pixel	N_r	Integer

The employed hardware acceleration techniques employed in Xilinx Vitis environment were the following: (a) the Split1/2 and Threshold arguments are passed to the kernel through separate wide buses (ports), (b) double wide ports are used for each one of Split1/2 and Threshold arguments in order to pass half of the argument values from each wide port in parallel, (c) the Split1/2, Threshold argument values are stored in local pairs of Block RAMs (BRAMs), (d) the loop that traverses the regression trees was split into two independent loops operating on different BRAMs of each pair, (e) pipeline and loop unrolling techniques are employed to the hardware kernel loops. The techniques (a)–(d) are shown in Figure 11.

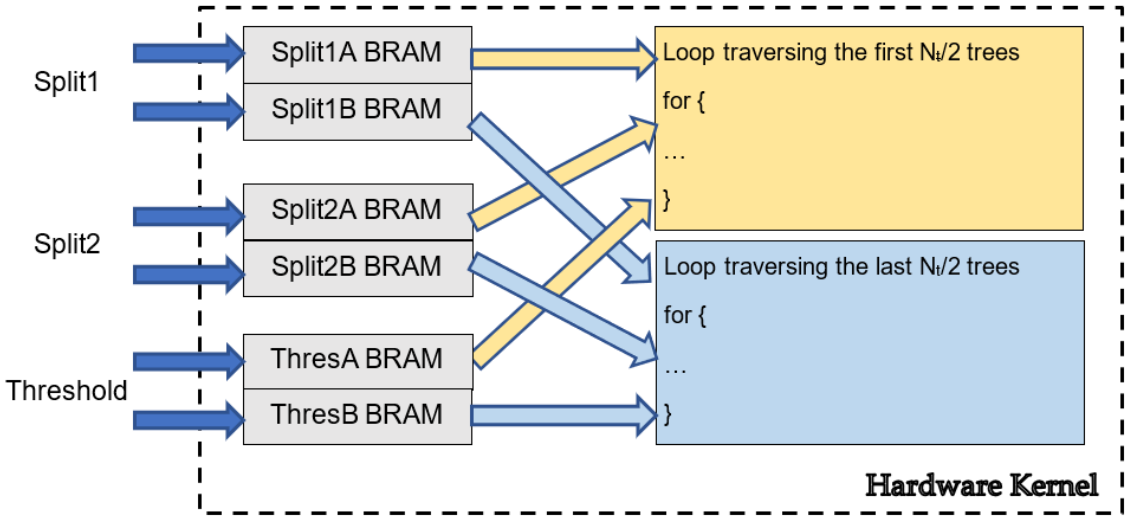


Figure 11. Hardware acceleration techniques employed for landmark prediction at ERT cascade stage level.

4. Experimental Results

In total, four ERT models have been trained using the training set of 270 images described in Section 3.1. These models differ in the number of cascade stages (T_c) and the number of regression trees (N_t) in each cascade stage, as described in Table 4. The specific T_c, N_t parameter values were selected to quantify the accuracy degradation if the number of cascade stages or regression trees is reduced compared to the default model M1 that is expected to achieve the maximum precision.

Table 4. ERT models trained.

Model	Cascade Stages T_c	Regression Trees N_t
M1 (default)	10	500
M2	10	400
M3	8	500
M4	8	400

The test set of P = 100 fish photographs has been derived from the 52 test photographs of the initial 322 image dataset, using LAE augmentation services. The error in the position of the landmarks is estimated from the comparison with the annotation defined as ground truth in the LAE editor. The relative error ε_{ri} between the estimated landmark $\hat{k}_i$ position and its corresponding position k_i in the ground truth annotation is the Euclidean distance between these two positions, expressed in pixels:

$$\varepsilon_{ri} = \|\hat{k}_i - k_i\|_2 \tag{11}$$

If $k_i = (w_i, h_i)$ and $\hat{k}_i = \left(\hat{w}_i, \hat{h}_i\right)$ and the image (width, height) is (w,h), the normalized relative error for landmark I, (ε_{ni}) is:

$$\varepsilon_{ni} = \sqrt{\frac{(\hat{w}_i - w_i)^2}{w^2} + \frac{\left(\hat{h}_i - h_i\right)^2}{h^2}} \tag{12}$$

The standard deviation (SD), σ_ε in the distribution of the landmark estimation error ε_{ni} across all L landmarks is:

$$\sigma_\varepsilon = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(\varepsilon_{ni} - \mu_\varepsilon)^2} \tag{13}$$

where μ_ε is mean error of ε_{ni} i.e.,

$$\mu_\varepsilon = \frac{1}{L}\sum_{i=1}^{L}\varepsilon_{ni} \tag{14}$$

The standard deviation in the distribution of estimation error ε_{nij} of a specific landmark i in P images ($0 \leq j < P$) is:

$$\sigma_i = \sqrt{\frac{1}{P}\sum_{j=1}^{P}\left(\varepsilon_{nij} - \mu_i\right)^2} \tag{15}$$

where μ_i is mean error of ε_{nij}, i.e.,

$$\mu_i = \frac{1}{P}\sum_{j=1}^{P}\varepsilon_{nij} \tag{16}$$

Another standard deviation metric (σ_P) used is in the average relative error $\mu_{\varepsilon j}$ (see Equation (14)) of all landmarks of an image in all the P test images:

$$\sigma_P = \sqrt{\frac{1}{P}\sum_{j=1}^{P}\left(\mu_{\varepsilon j} - \mu_P\right)^2} \tag{17}$$

where μ_P is the mean of $\mu_{\varepsilon j}$:

$$\mu_P = \frac{1}{P}\sum_{j=1}^{P}\mu_{\varepsilon j} \tag{18}$$

Table 5 shows the average, minimum and maximum, absolute and relative errors that have appeared in all landmarks and all the P test images when the model M1 is employed for maximum accuracy.

Table 5. Global absolute and relative errors for M1.

Type of Error	Error
Min Absolute Error	0.36 pixels
Max Absolute Error	78.33 pixels
Average Absolute Error	17.54 pixels
Min Relative Error	0.1%
Max Relative Error	33.6%
Average Relative Error	4.8%

The standard deviation σ_ε limits as well as the σ_P deviation of the average error in the P test images are listed in Table 6.

Table 6. Relative error standard deviation σ_ε limits and σ_P for M1.

Parameter	Value
Min σ_ε deviation	0.0068
Max σ_ε deviation	0.081
Average σ_ε deviation	0.03
σ_P deviation	0.0215

The mean error μ_i of each landmark i, along with its standard deviation σ_i is plotted in Figure 12, for model M1. This plot is of particular interest because it highlights the landmarks that show the highest error.

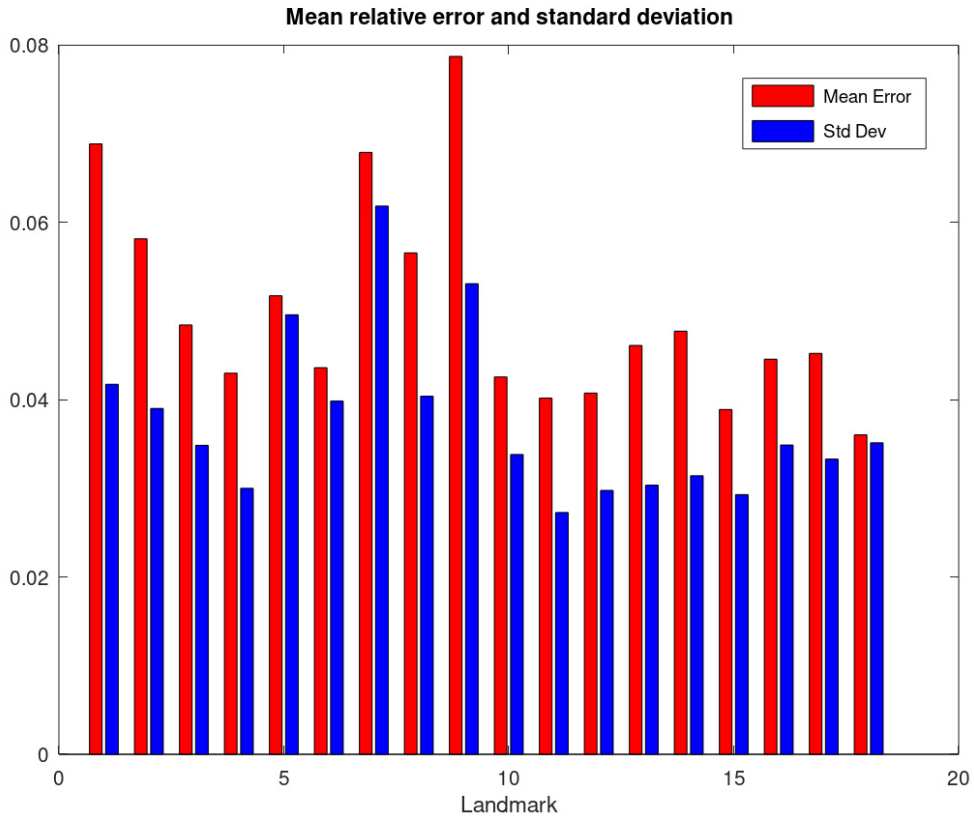


Figure 12. The mean relative error and standard deviation for each one of the 18 landmarks.

The error in the relative height and length estimation for the default ERT model M1 of the fish in the test set is listed in Table 7 along with their standard deviations.

Table 7. Error and standard deviation in measuring relative fish length and height.

Parameter	Value
Fish length error	5.4%
Length error deviation	0.049
Fish height error	4.5%
Width error deviation	0.062

Figure 13 can be used to compare the error shown by the ERT models of Table 4, in the estimation of the fish length, height and the location of the eyes and gills from the corresponding landmarks.

Concerning the fish orientation methods described in Section 3.4, Table 8 lists the success rates achieved with each method. The PCA method is capable of recognizing the fish tilt with a very good accuracy (less than $\pm 10^{\circ}$ in more than 95% of the cases). However, the direction that the fish is facing is recognized with much lower accuracy as shown in Table 8. COD performs a draft classification in left or right direction, while FOD performs a more detailed orientation classification in quadrants Q0–Q3 with the success rate listed in Table 8. Eye template matching (TM) is also used to classify the direction in one of the Q0–Q3 quadrants. A number of combinations of these orientation classification methods are then tested.

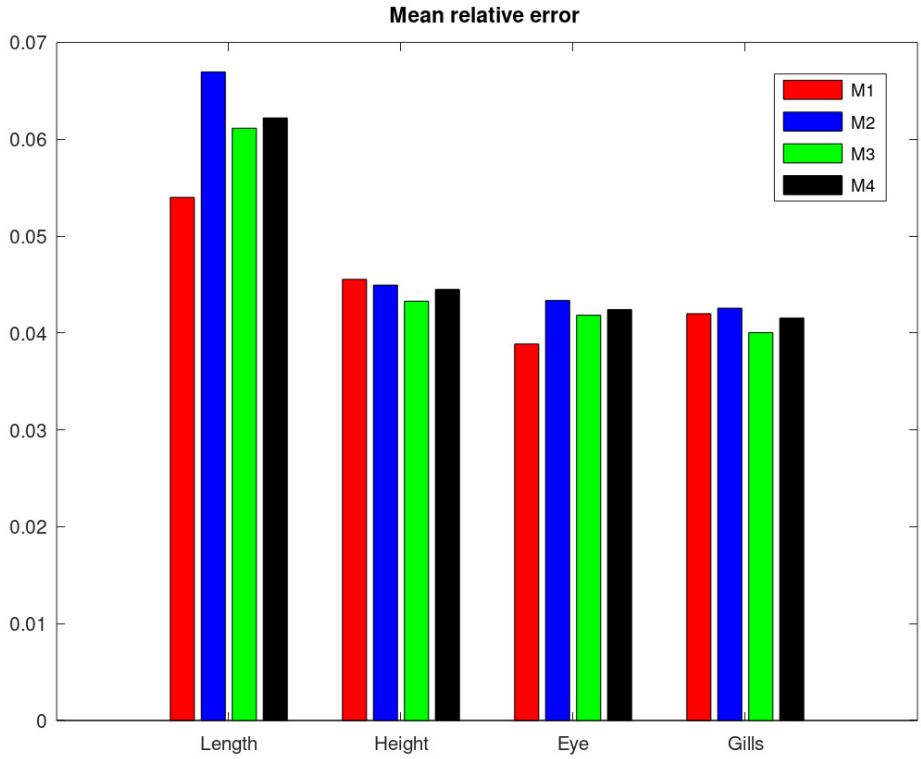


Figure 13. The mean relative error for estimating the fish length, height, eye and gills position with each one of the M1–M4 ERT models.

Table 8. Success rate in fish direction recognition with PCA, left or right direction classification with COD, classification in Q0–Q3 quadrants with FOD, template matching (TM) and their combinations.

	PCA	**COD**	**FOD**	**TM**	**PCA + COD**	**PCA + COD (2)**	**PCA + TM**
Success Rate:	44.8%	67.2%	43.1%	63.8%	65.5%	65.5%	77.6%

In PCA + COD, the tilt found by PCA is used while COD is used to detect the direction. In PCA + COD (2), COD direction is taken into consideration only if the confidence is above a threshold. In PCA + TM, the coarse left–right direction indicated by TM is taken into consideration to decide the direction on the tilt estimated by PCA. If, for example, the tilt is from bottom-left to top-right, then Q1 is selected if TM finds the fish eye in the right quadrants (Q1, Q3). If the fish eye is found in the left quadrants (Q0, Q2), then Q2 is selected. In PCA + TM (2) method, the TM direction is considered only if the template matching found the fish eye in one of the quadrants that are compatible with the fish tilt indicated by PCA. For example, if the fish is facing up-right, its caudal fin is in Q2 and its head is in Q1. If the TM method finds the fish eye in Q1 or Q2 then, the direction indicated by TM will be assumed correct. Specifically, with fish eye found by TM in Q1 it will correctly be recognized that the fish is facing up-right while if the fish eye is found in Q2 it will be assumed by mistake that the fish is facing down-left. If the TM finds the fish eye in Q0 or Q3, the direction indicated by TM will not be taken into consideration and only the direction indicated by PCA will be used.

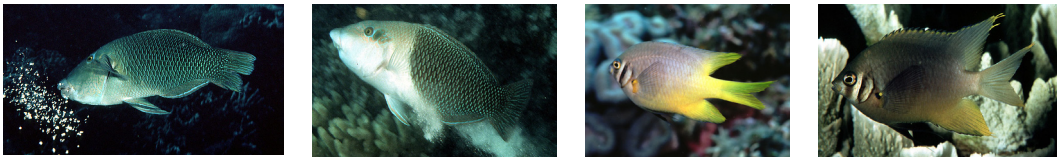
In Table 9, a comparison can be found between our fish length and height estimation method and the references that present fish size estimation results. The last column of Table 9 lists the frame-processing latencies (D_f) of the referenced approaches and our work.

Table 9. Fish size estimation error and frame-processing speed comparison.

Reference	Description	Error	Frame Processing Latency D_f
[21]	Tuna fish size estimation	SD: 0.328–0.396	2 s
[26]	Fish length estimation	Error 5%	Nor reported
[27]	Fish size estimation	Error 8%	0.2 s (NVIDIA ® Tesla K40)
[32]	Fish length estimation	Error 2–8% depending on the fish size	
Previous work [10]	Fish length estimation	Error 4.93%	2.7–5.1 s (Intel i5 platform)
Previous work [10]	Fish height estimation	Error 10.13%	
This work	Fish length estimation	Error 5.4% SD: 0.049	<0.5 μs (on Intel i5 platform)
This work	Fish height estimation	Error 4.5% SD: 0.062	<16 ms (on Xilinx ZCU102 platform)

5. Discussion

The error in the landmark position estimation as presented in Tables 5, 7 and 9 and Figures 11 and 12 is largely due to the low contrast of the images in the employed dataset [34]. Other referenced approaches [23–25], are also tested with low-quality underwater images. However, in most cases they display fish that are more clearly visible than the images in the UVIMEF dataset. The fish in images from ImageCLEF/LifeCLEF dataset used in [23], Fish4Knowledge [24] and ImageNet [25] are more distinct as shown in the example photographs of Figure 14 (they can be compared to sample images from UVIMEF in Figure 7b–e).



(**a**) (**b**) (**c**) (**d**)

Figure 14. Sample images from ImageCLEF (**a**,**b**) and Fish4Knowledge (**c**,**d**) datasets.

To measure the contrast in the images of a dataset, various metrics can be used. The Root Mean Square (RMS) contrast in an image Im, with $Row \times Col$ pixels is defined as:

$$RMS\ contrast = \sqrt{\frac{1}{Row \cdot Col} \sum_{i=1}^{Row} \sum_{j=1}^{Col} \left(Im - \widetilde{Im}\right)^2} \tag{19}$$

where $\widetilde{Im}$ is the average intensity of the pixels in image Im. Another popular metric is the Michelson contrast that is based on the minimum (I_{min}) and maximum (I_{max}) pixel intensities in an image Im:

$$Michelson\ contrast = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{20}$$

The entropy of an image or of a specific region in an image measures the information incorporated is this region. Thus, entropy is also related with contrast since higher entropy indicates more information expressed as abrupt changes in the intensity of neighboring pixels. Entropy is defined as:

$$Entropy = -\sum P_e. * log_2 P_e \quad (21)$$

P_e contains the normalized histogram counts of the image *Im.* A total of 20 indicative photographs have been selected from our dataset (UVIMEF) and the same number of images from ImageCLEF and Fish4Knowledge datasets. The average values of the contrast metrics defined above are listed in Table 10. As can be seen from this table our dataset has the lowest contrast. UVIMEF has a much smaller RMS and Michelson contrast than the other two datasets. Concerning the entropy, only Fish4Knowledge has lower average entropy than UVIMEF. Moreover, the fish dimensions in UVIMEF photographs are quite small resulting in patches that may have extremely low resolution (e.g., 70 × 30 pixels). The low resolution and contrast of the images that serve as input to our shape alignment approach, pose much worse conditions for our experiments, compared with the referenced approaches.

Table 10. Contrast comparison.

Dataset	RMS Contrast	Michelson Contrast	Entropy
UVIMEF used in this work	23.61	0.82	6.80
ImageCLEF used in [23]	63.26	1.00	7.37
Fish4Knowledge used in [24]	55.64	0.94	6.10

From the experimental results presented in the previous section, the average relative error (using M1 model) in the alignment of a single landmark is 4.8% (corresponding to an absolute error of 17.54 pixels) with the following SDs: σ_ε = 0.03, σ_P = 0.0215. The relative error in the estimation of fish length is 5.4% and 4.5% in the estimation of the height (with the corresponding SDs being 0.049 and 0.062, respectively). Taking into consideration that the length of the fish recognized in the photographs of the dataset ranges from 10 cm to 30 cm, the average absolute error is in the order of 0.5 cm–1.5 cm.

More details concerning the accuracy in the alignment of individual landmarks can be found in Figure 12. Specifically, landmarks 7 (top of the caudal fin) and 9 (bottom of the caudal fin) are located with a mean error equal to 6.8% and 7.8%, respectively. These are the highest relative errors measured per landmark. They appear at the landmarks that mark the edge of the caudal fin because in most photographs of the UVIMEF dataset used for training and testing, the caudal fin perimeter is often indistinguishable from the background. Landmarks 1 (mouth) and 8 (middle of the caudal fin), that are used for the estimation of fish length are located with a mean error of 6.8% and 5.6%, respectively. When they are combined to estimate the fish length, the average relative error is 5.4%.

Landmarks 3 and 13 are used to estimate fish height. Their average relative error is 4.8% and 4.6%, lower than the error shown by landmarks 1 and 8 that are used for length estimation. For this reason, the error in fish height estimation (4.5%) is lower than that of the fish length (5.4%). Other landmarks of interest are No. 17 and 18 that are used to locate the fish eye ROI. These landmarks are located with an average relative error of 4.5% and 3.6%. Taking into consideration the fish size range mentioned above, this relative error in the fish eye localization is interpreted to about 0.4 cm–1.2 cm. In the experiments conducted, the fish eye was not always found between landmarks No. 17 and 18. Nevertheless, additional pattern recognition methods can be applied to localize the exact position of the eye in the neighborhood of these landmarks. Similarly, the position of the gills is another ROI located by landmarks No. 14, 15 and 16. The mean relative errors in the estimation of the position of these landmarks range between 3.8% and 4.7%.

In Figure 13, the relative error in the estimation of four morphological features, by the ERT models listed in Table 4, is displayed. More specifically, the error in the estimation of the fish length, height, as well as the position of the eyes and gills is compared. In all cases, the error of model M2 is slightly higher than that of model M3 and the error of M3 is slightly higher than the error M4. However, the error shown by the default model M1 is higher than that of M3 in the estimation of the fish height and the position of the fish gills. Model M3 seems to show an error comparable to that of M1 and can replace it, if higher processing speed is required. If the frame-processing latency of the default model M1 (N_t = 500, T_c = 10) is D_f, the following equation estimates the latency D'_f of a different model with N'_t trees and T'_c cascade stages:

$$D'_f = \frac{T'_c N'_t}{T_c N_t} D_f \tag{22}$$

For example, the latency of M3 ($N'_t = 500,\ T'_c = 8$) is $D'_f = \frac{8 \cdot 500}{10 \cdot 500} D_f = 0.8 \cdot D_f$. M4 ($N'_t = 400,\ T'_c = 8$) is the model with the lowest latency: $D'_f = \frac{8 \cdot 400}{10 \cdot 500} D_f = 0.64 \cdot D_f$. Thus, M4 is expected to have 56.25% higher speed than M1.

Concerning the fish orientation classification, Table 8 shows that the best results are achieved with the combination of PCA with TM when the false eye template matching estimations are ignored. PCA is capable of detecting the tilt of the fish with high accuracy. However, it could detect the direction of the fish in only 44.8% of the cases, using the low-contrast images of the UVIMEF dataset. The COD achieved a higher success rate (67.2%) but can detect only a draft left or right direction. The FOD method could be used to classify the direction of the fish in four quadrants but its classification accuracy is only 43.1%. On the other hand, fish eye template matching has a relatively higher success rate of 63.8%. This could have been even higher, if the resolution and the quality of the dataset images were better because in many fish image patches the eye is not visible at all. In these cases, the eye is confused either with background objects or with other parts of the fish like a strip in the caudal fin of some species such as Diplodus annularis (see Figure 7). Certain combinations of these orientation detection methods were also tested as shown in Table 8. Combining PCA with the left or right classification of COD achieved a success rate of 65.5%. The highest accuracy was achieved when the PCA was combined with TM and can reach 79.3%. A much higher orientation accuracy is expected to be achieved if the track of the fish is also taken into consideration as explained in Section 3.5.

Comparing the fish size estimation methods listed in Table 9 as well as the errors displayed in Figure 13, it is obvious that the proposed fish length or height estimation achieves one of the best accuracies reported for morphological feature estimation. In [26], a slightly lower error (5%) is achieved in the estimation of the fish length while in [32] the error is lower only in some specific fish sizes. However, in most of the cases presented in the literature, fish size is estimated in a controlled environment (e.g., on a conveyor belt) or with high-resolution underwater images and clearly visible fish as described in Figure 14 and Table 10. Estimating fish size in low-contrast and resolution images like those generated from UVIMEF dataset, is a much more challenging task. It is obvious that all the errors listed in Tables 5–7 and 9, as well as Figures 12 and 13 would be lower if the ERT models had been trained with higher-quality images. It is also worth noting from Table 9, that the accuracy in the present work is much better compared to previous work [10]. The frame-processing speed of the current approach is also orders of magnitude higher than that of the previous work [10].

In summary, the developed framework offers a number of useful services for fish monitoring such as morphological feature estimation, fish orientation classification and fish tracking. These services can be employed both for monitoring fish in free waters and aquacultures. The fish detection, orientation classification and shape alignment methods for morphological feature estimation were described in detail. The principles of fish tracking in the developed framework were also discussed.

One of the limitations of the current work is the latency of the fish detection. Specific directions were given to pipeline the fish detection in specific frames with other tasks that can run as parallel threads. These tasks can be the bounding box interpolation in intermediate frames between actual fish detections, the execution of the orientation classification and the shape alignment. Hardware acceleration of the shape alignment process was applied for embedded target platforms. Similarly, the inference for fish detection can also be implemented in hardware on the same target platform. Developing such an architecture is part of our on-going work in order to achieve a high frame-processing speed for the overall system and support real-time operation. Finally, more sophisticated techniques can also be incorporated into the presented fish-tracking approach. For example, feedback from the shape alignment stage can be exploited to identify with higher confidence the fish in successive frames without confusing their positions. The fish orientation can also indicate when the fish is changing direction in its track.

6. Conclusions

Fish detection, orientation classification, size estimation, locating regions of interest and fish tracking are supported in the framework presented in this paper. It can be exploited for fish monitoring in aquaculture systems and open sea. It is based on deep learning for fish detection, OpenCV services for orientation classification and the adaptation of a shape alignment machine learning method called Ensemble of Regression Trees. Hardware and software acceleration techniques have been developed for the shape alignment process achieving a frame-processing latency of less than 0.5 μs on an Intel i5 platform or less than 16 ms on an embedded platform with programmable logic. The fish detection is performed with an accuracy higher than 95% since almost all of the fish that were expected to be monitored, were detected. The orientation of the fish is classified in four major directions with 80% success rate. The relative fish size estimation was also performed with an accuracy ranging between 4.5% and 5.5%. Preliminary demos and tutorials for this work are available as Supplementary Materials.

Future work will focus on employing hardware acceleration for the fish detection inference, in order to improve the frame-processing speed of the overall system. Moreover, the employed fish-tracking method will be implemented taking into consideration additional information such as the speed and the changes in the direction of the fish.

Supplementary Materials: A description of how DEST was ported to Ubuntu environment and MS Visual Studio 2019 as well as the use of LAE editor can be found in the videos of the playlist https://www.youtube.com/playlist?list=PLXUuQj2gQ4s9TXI12kP9WMaxthWrjAweF. The description of how hardware acceleration has been applied to DEST video tracking application for face shape alignment can be found in https://www.youtube.com/watch?v=OlCBCwroAkw&ab_channel=ESDALab (accessed on 1 August 2023).

Author Contributions: Conceptualization, N.P. and G.K.; methodology, N.P.; software, N.P.; validation, N.P. and G.K.; resources, C.P.A. and N.V.; data curation, N.P.; writing—original draft preparation, N.P.; writing—review and editing, all; supervision, N.V.; project administration, N.P. and N.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: UVIMEF Dataset: https://www.kaggle.com/datasets/nikospetrellis/uvimef (accessed on 1 August 2023).

Acknowledgments: The authors wish to thank the student Spilios Kostopoulos for conducting part of the experiments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

BMA	Binary Mask Annotation
BRAM	Block RAM
CLEF	Cross Language Evaluation Forum
CNN	Convolution NN
COD	Coarse Orientation Detection
CPU	Central Processing Unit
DCNN	Deep CNN
DEST	Deformable Shape Tracking
DIDSON	Dual Frequency Identification Sonar
ERT	Ensemble of Regression Trees
FPGA	Field Programmable Gate Array
FPI	Fish Position Interpolation
FOD	Fine Orientation Detection
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
LAE	Landmark Annotation Editor
ML	Machine Learning
NN	Neural Network
PCA	Principal Component Analysis
PM	Pattern Matching
R-CNN	Region-based CNN
RMS	Root Mean Square
ROI	Region of Interest
SCIA	Segmented Color Image Annotation
SD	Standard Deviation
SVD	Singular Vector Decomposition
TM	Template Matching
UVIMEF	Underwater Videos–Images of Mediterranean Fish
UWP	Universal Windows Platform
YOLO	You Only Look Once

References

1. Vo, T.T.E.; Ko, H.; Huh, J.-H.; Kim, Y. Overview of Smart Aquaculture System: Focusing on Applications of Machine Learning and Computer Vision. *Electronics* **2021**, *10*, 2882. [CrossRef]
2. Zion, B. The use of computer vision technologies in aquaculture—A review. *Comput. Electron. Agric.* **2012**, *88*, 125–132. [CrossRef]
3. Mathiassen, J.R.; Misimi, E.; Bondø, M.; Veliyulin, E.; Østvik, S.O. Trends in application of imaging technologies to inspection of fish and fish products. *Trends Food Sci. Technol.* **2011**, *22*, 257–275. [CrossRef]
4. Fish Detection. Available online: https://github.com/kwea123/fish_detection (accessed on 1 March 2023).
5. OpenCV. Available online: https://opencv.org/ (accessed on 25 May 2023).
6. Kazemi, V.; Sullivan, J. One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 867–1874. [CrossRef]
7. Dlib C++ Library. Available online: http://dlib.net/ (accessed on 25 May 2023).
8. Deformable Shape Tracking (DEST). Available online: https://github.com/cheind/dest (accessed on 25 May 2023).
9. Petrellis, N.; Christakos, P.; Zogas, S.; Mousouliotis, P.; Keramidas, G.; Voros, N.; Antonopoulos, C. Challenges Towards Hardware Acceleration of the Deformable Shape Tracking Application. In Proceedings of the 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), Singapore, Singapore, 4–7 October 2021. [CrossRef]
10. Petrellis, N. Measurement of Fish Morphological Features through Image Processing and Deep Learning Techniques. *Appl. Sci.* **2021**, *11*, 4416. [CrossRef]
11. Gu, K.; Zhai, G.; Yang, X.; Zhang, W.; Chen, C.W. Automatic Contrast Enhancement Technology with Saliency Preservation. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 1480–1494. [CrossRef]
12. Gu, K.; Lin, W.; Zhai, G.; Yang, X.; Zhang, W.; Chen, C.W. No-Reference Quality Metric of Contrast-Distorted Images Based on Information Maximization. *IEEE Trans. Cybern.* **2017**, *47*, 4559–4565. [CrossRef]
13. Franceschelli, L.; Berardinelli, A.; Dabbou, S.; Ragni, L.; Tartagni, M. Sensing Technology for Fish Freshness and Safety: A Review. *Sensors* **2021**, *21*, 1373. [CrossRef]

14. Freitas, J.; Vaz-Pires, P.; Câmara, J.S. From aquaculture production to consumption: Freshness, safety, traceability and authentication, the four pillars of quality. *Aquaculture* **2020**, *518*, 734857. [CrossRef]
15. Choi, J.W.; Jang, M.K.; Hong, C.W.; Lee, J.W.; Choi, J.H.; Kim, K.B.; Xu, X.; Ahn, D.H.; Lee, M.K.; Nam, T.J. Novel application of an optical inspection system to determine the freshness of Scomber japonicus (mackerel) stored at a low temperature. *Food Sci. Biotechnol.* **2020**, *29*, 103–107. [CrossRef]
16. Dowlati, M.; Mohtasebi, S.S.; Omid, M.; Razavi, S.H.; Jamzad, M.; De La Guardia, M. Freshness assessment of gilthead sea bream (Sparus aurata) by machine vision based on gill and eye color changes. *J. Food Eng.* **2013**, *119*, 277–287. [CrossRef]
17. Li, X.; Shang, M.; Qin, H.; Chen, L. Fast accurate fish detection and recognition of underwater images with Fast R-CNN. *Oceans* **2015**, *2015*, 1–5. [CrossRef]
18. Jalal, A.; Salman, A.; Mian, A.; Shortis, M.; Shafait, F. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecol. Inform.* **2020**, *57*, 101088. [CrossRef]
19. Sung, M. Vision based real-time fish detection using convolutional neural network. *Oceans* **2017**, 1–6. [CrossRef]
20. Xie, Y. Improved Gaussian Mixture Model in Video Motion Detection. *J. Multimed.* **2013**, *8*, 527–533. [CrossRef]
21. Lekunberri, X.; Ruiz, J.; Quincoces, I.; Dornaika, F.; Arganda-Carreras, I.; Fernandes, A.A. Identification and measurement of tropical tuna species in purse seiner catches using computer vision and deep learning. *Ecol. Inform.* **2022**, *67*, 101495. [CrossRef]
22. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. Available online: https://arxiv.org/abs/1703.06870 (accessed on 25 May 2023).
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
24. Qin, H.; Li, X.; Liang, J.; Peng, Y.; Zhang, C. Deepfish: Accurate underwater live fish recognition with a deep architecture. *Neurocomputing* **2016**, *187*, 49–58. [CrossRef]
25. Sun, X.; Shi, J.; Dong, J.; Wang, X. Fish recognition from low-resolution underwater images. In Proceedings of the 9th IEEE International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 471–476, Datong, China, 15–17 October 2016. [CrossRef]
26. Ubina, N.A.; Cheng, S.C.; Chang, C.C.; Cai, S.Y.; Lan, H.Y.; Lu, H.Y. Intelligent underwater Stereo Camera Design for Fish Metric Estimation Using Reliable Object Matching. *IEEE Access* **2022**, *10*, 74605–74619. [CrossRef]
27. Fisher, M.H.; French, J.; Gorpincenko, A.; Mackiewicz, M.; Holah, H.; Clayton, L.; Selkow, R. Motion Stereo at Sea: Dense 3D Reconstruction from Image Sequences Monitoring Conveyor Systems on Board Fishing Vessels. *IET Image Process.* **2022**, *17*, 349–361. [CrossRef]
28. Karnani, K.; Pepper, J.; Bakis, Y.; Wang, X.; Bart, H.; Breen, D.; Greenberg, J. Computational Metadata Generation Methods for Biological Specimen Image Collections. 27 April 2022, PREPRINT (Version 1) Available at Research Square, European Bioinformatics Institute. Available online: https://www.researchsquare.com/article/rs-1506561/v1 (accessed on 1 August 2023).
29. Kandimalla, V.; Richard, M.; Smith, F.; Quirion, J.; Torgo, L.; Whidden, C. Automated Detection, Classification and Counting of Fish in Fish Passages with Deep Learning. Front. *Mar. Sci.* **2022**, *8*, 823173. [CrossRef]
30. Alori, J.; Descoins, A.; Ríos, B.; Castro, A. Norfair Library. Tryolabs/Norfair: v0.3.1. Available online: https://zenodo.org/record/5146254 (accessed on 25 May 2023).
31. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
32. Martignac, F.; Daroux, A.; Bagliniere, J.L.; Ombredane, D.; Guillard, J. The use of acoustic cameras in shallow waters: New hydroacoustic tools for monitoring migratory fish population. A review of DIDSON technology. *Fish Fish.* **2015**, *16*, 486–510. [CrossRef]
33. Terayama, K.; Shin, K.; Mizuno, K.; Tsuda, K. Integration of sonar and optical camera images using deep neural network for fish monitoring. *Aquac. Eng.* **2019**, *86*, 102000. [CrossRef]
34. Petrellis, N.; Keramidas, G.; Antonopoulos, C.P.; Voros, N. UVIMEF [Data Set]. Kaggle. 2023. Available online: https://www.kaggle.com/datasets/nikospetrellis/uvimef (accessed on 25 May 2023).
35. Chyrka, I.; Kharchenko, V. 1D direction estimation with a YOLO network. In Proceedings of the 2019 European Microwave Conference in Central Europe (EuMCE), Prague, Czech Republic, 13–15 May 2019; pp. 358–361.
36. Hara, K.; Vemulapalli, R.; Chellappa, R. Designing Deep Convolutional Neural Networks for Continuous Object Orientation Estimation. *arXiv* **2017**, arXiv:1702.01499. [CrossRef]
37. Song, F.; Guo, Z.; Mei, D. Feature selection using principal component analysis. In Proceedings of the 2010 International Conference on System Science, Engineering Design and Manufacturing Informatization, Yichang, China, 18 November 2010; pp. 27–30. [CrossRef]
38. De Silva, A. Object Orientation Detection and Correction Using Computer Vision. Culminating Projects in Computer Science and Information Technology. 33. St. Cloud State University. 2020. Available online: https://repository.stcloudstate.edu/csit_etds/33 (accessed on 25 May 2023).
39. Lendave, V. Detecting Orientation of Objects in Image Using PCA and OpenCV. Available online: https://analyticsindiamag.com/detecting-orientation-of-objects-in-image-using-pca-and-opencv/ (accessed on 15 May 2023).

40. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Sys. Man. Cyber* **1979**, *9*, 62–66. [CrossRef]
41. Eigen 3.3.9. Available online: https://eigen.tuxfamily.org/ (accessed on 15 May 2023).