

An Intelligent and Cost-effective Remote Underwater Video Device for Fish Size Monitoring

Gianpaolo Coro^{a,1,2,*}, Matthew Bjerregaard Walsh^b

^a*Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR, Pisa, Italy*

^b*Food and Agriculture Organization of the United Nations, Viale delle Terme di Caracalla, 00153 Rome, Italy*

Abstract

Monitoring the size of key indicator species of fish is important to understand ecosystem functions, anthropogenic stress, and population dynamics. Standard methodologies gather data using underwater cameras, but are biased due to the use of baits, limited deployment time, and short field of view. Furthermore, they require experts to analyse long videos to search for species of interest, which is time consuming and expensive. This paper describes the Underwater Detector of Moving Object Size (UDMOS), a cost-effective computer vision system that records events of large fishes passing in front of a camera, using minimalistic hardware and power consumption. UDMOS can be deployed underwater, as an unbaited system, and is also offered as a free-to-use Web Service for batch video-processing. It embeds three different alternative large-object detection algorithms based on deep learning, unsupervised modelling, and motion detection, and can work both in shallow and deep waters with infrared or visible light.

Keywords: Computer Vision, Biodiversity Conservation, Fish Size, Baited Remote Underwater Video, Artificial Intelligence, Deep Learning, Unsupervised Modelling, Motion Detection

*Corresponding author

Preprint submitted to *Ecological Informatics* (Gianpaolo Coro), matthew.walsh@fao.org May 4, 2021
(Matthew Bjerregaard Walsh)

¹Telephone Number: +39 050 315 8210

²Fax Number: +39 050 621 3464

11 **1. Introduction**

12 Monitoring the frequency of detection of key indicator species of marine fishes in
13 their native habitat is a useful method of gathering data to understand characteristics such
14 as population distribution, biomass change, anthropogenic impact, and the function of
15 ecosystem relationships such as mutualistic behaviour. Common approaches to gathering
16 such data use standalone video recording devices, sometimes equipped with baits that are
17 deployed underwater (baited remote underwater video device, BRUV), capturing footage
18 continuously until on-board storage is exhausted (Cappo et al., 2004; Vos et al., 2014; Mal-
19 let and Pelletier, 2014). This continuous recording results in a number of characteristics
20 which produce bias in data collection, e.g. the duration of deployment time capturing data
21 is limited to a few hours and this limitation also encourages the use of non-passive baited
22 camera systems, which may affect inference regarding the presence of species. Upon re-
23 trieving the devices, experts need to view single time period samples in the form of long
24 videos to search for species of interest and conduct further analysis such as taxonomic
25 confirmation, maximum abundance of fish per frame (MaxN), and estimation of life stages
26 dependant upon their size. In some cases, dual cameras are used to allow on-screen mea-
27 sures of fish length by helping expert-observation with shape analysis tools (Costa et al.,
28 2006).

29 In recent years, researchers have applied artificial intelligence to accelerate the post
30 data capture process (Qin et al., 2016; Shafait et al., 2016; Marini et al., 2018; González-
31 Rivero et al., 2020). However the methodology for video collection of information, and
32 port capture data retrieval from video and its analysis is both time consuming, expen-
33 sive, and generates substantial video data. Another practical difficulty is to find, hire, and

34 train professional video operators. Today, operators are mostly university students whose
35 commitment time, interest, and availability is often limited and fragmented. Also, this
36 approach offers no alternative to the continuous recording approach. On the contrary, an
37 automatic solution - for example an edge-computing vision system - could autonomously
38 monitor for far longer time underwater while constantly capturing video event data. Al-
39 though analogue solutions for motion sensing may offer alternatives to a passive AI-based
40 approach (Daum, 2005; Spampinato et al., 2008; Hsiao et al., 2014; Salman et al., 2020),
41 many have implications regarding bias and limited capacity underwater (e.g. microwave
42 motion sensors) (Yoon et al., 2012; Hussey et al., 2015). Also, detection methods may
43 perturb the presence or absence of species (e.g. analogue sonar scanning) or simply are
44 not precise enough to discriminate between debris or smaller organisms. With these con-
45 siderations in mind, in order to create a detection system for larger indicator species, such
46 as sharks and rays, a solution is needed to detect animated moving objects and classify
47 them in terms of their size.

48 This paper describes the Underwater Detector of Moving Object Size (UDMOS) soft-
49 ware, a cost-effective computer vision system that can be deployed underwater and is able
50 to identify and record videos of fishes of large size moving in front of a camera. The min-
51 imum fish size to detect is a configurable parameter that defines the minimum percentage
52 object's size with respect to the camera's frame size that should trigger video recording.
53 UDMOS can work in shallow as well as deep waters, and uses minimal hardware and de-
54 ployment equipment to operate. Hardware is scalable from a very inexpensive solution,
55 based only on a single IR camera and one Raspberry Pi4 device, to more expensive solu-
56 tions that use more cameras and more powerful hardware. UDMOS can be deployed to

57 capture the presence of large fishes over long time periods and this characteristic reduces
58 the need for a baited system to concentrate activity in front of the camera. This non-
59 invasive solution is cost effective and reduces the bias for fish detection that the presence
60 of bait may cause. Additionally, UDMOS is offered as a free-to-use cloud Web service
61 that can post-process videos captured by standard BRUVs.

62 In this study, the performance of the workflow is assessed on five real operational
63 cases under different light and depth conditions, using different object detection algo-
64 rithms that can work on minimal hardware. The performance comparison also involves
65 a movement-detection algorithm embedded in UDMOS. Overall, UDMOS addresses the
66 following research question: *Can we use modern single-board computers to design a large,*
67 *precise, non-invasive, efficient, and cost-effective system for monitoring fish of a specific*
68 *size range?*

69 UDMOS embeds several novel features with respect to alternative remote underwater
70 video devices (Edgington et al., 2006; Schlining and Stout, 2006; Ebner et al., 2014; Codd-
71 Downey et al., 2017). Internally, it can use one among three different approaches to detect
72 objects or movement. These approaches work on a low-resource hardware, with different
73 response times. One advantage of UDMOS with respect to other approaches (Brooks et al.,
74 2011; Struthers et al., 2015; Quevedo et al., 2017; Schmid et al., 2017), is that it can work
75 with one basic camera to estimate the approximate distance of an object. Different from
76 other solutions (Harvey et al., 2003; Edgington et al., 2006; Schlining and Stout, 2006;
77 Van Damme, 2015; Palazzo et al., 2014), UDMOS is conceived to automatically adapt to
78 both low and high power hardware. Unlike camera-trap systems that use motion detec-
79 tion (Zhou et al., 2008; Kays et al., 2010; Miguel et al., 2016; Apps and McNutt, 2018;

80 Golkarnarenji et al., 2018; Marcot et al., 2019), our workflow improves motion detection
81 precision through an adaptive thresholding algorithm and offers alternative object detec-
82 tion models. Similar to other underwater devices (Edgington et al., 2003; Hermann et al.,
83 2020), UDMOS can work in both IR and visible light conditions by automatically select-
84 ing the optimal configuration based on the scene brightness. The workflow is also able to
85 approximately account for common issues found by other systems due to small and close
86 fishes attracted by the recording device (Harvey et al., 2007; Dunlop et al., 2015; Coghlan
87 et al., 2017). Overall, UDMOS strongly reduces the amount of irrelevant video data pro-
88 duced, especially when the events to capture are rare, and thus is beneficial both in terms
89 of human time saving and hardware costs. It can be used to implement an edge computing
90 as well as an as-a-service batch processing system for current BRUV systems.

91 UDMOS can be coupled with modern underwater species identification systems that
92 work on BRUV-collected videos. Both open-source (Dawkins et al., 2017) and commercial
93 fish identification software and abundance estimators (Santana-Garcon et al., 2014) exist
94 that can work on BRUV videos, and thus on the UDMOS videos. These systems usually
95 work offline and are used after the underwater video capture session. Indeed, they have
96 hardware requirements that are not affordable by current low-cost and embeddable tech-
97 nology. Most species identification systems and abundance estimators are based on deep
98 learning models (Dawkins et al., 2017), which have demonstrated optimal performance
99 with respect to other alternative models (Sheaves et al., 2020) and can over-perform human
100 classification on species-specific identification tasks (Konovalov et al., 2019; Knausgård
101 et al., 2020; Sheaves et al., 2020). However, these model are still unreliable in species
102 abundance estimation and generally cannot substitute human experts (Connolly et al.,

103 2021). Furthermore, they have demanding hardware requirements - e.g., 64-bit Operat-
104 ing Systems and powerful GPUs (Dawkins et al., 2017) - and thus are normally provided
105 as-a-service through high-performance computing architectures that maximise both their
106 efficiency and effectiveness (Candela et al., 2016; Coro et al., 2018; Sheaves et al., 2020).
107 Overall, on-board species identification requires powerful and expensive hardware and
108 battery capacity - GPU processing is very power-consuming, and 64-bit Operating Sys-
109 tems on edge computers are still at an early stage - and thus on-board processing is usually
110 limited to motion detection (Sheehan et al., 2020).

111 UDMOS is principally conceived to reduce the time that either human experts or au-
112 tomatic models require to post-process the captured underwater videos for species recog-
113 nition, size measurement, and abundance estimations. Thus, one crucial requirement is
114 that its performance on an edge computer is optimal as it would be on a cloud computing
115 platform. For this reason, UDMOS addresses the simpler task of triggering the recording
116 when a large fish passes in front of the camera rather than recognizing the fish. This choice
117 has the advantage to (i) avoid biases due to misclassification, (ii) be applicable to a large
118 spectrum of species (i.e. not only those for which a model was trained), and (iii) reduce
119 recorded video length to a much lower length than the continuous recording (Section 3).

120 The next sections describe the complete UDMOS workflow and target hardware (Sec-
121 tion 2) and its effectiveness and efficiency (Section 3). Finally, a discussion of the results
122 and the potential applications of UDMOS is reported (Section 4).

123 **2. Material and Methods**

124 In this section, the general architecture of the UDMOS workflow is described by fol-
125 lowing the flowchart in Figure 1.

126 *2.1. Hardware Requirements*

127 The workflow was designed to facilitate a minimal hardware configuration and to con-
128 sume low power. Our minimal target hardware was a Raspberry Pi-4 Model B 4GB
129 equipped with ARM-Cortex-A72 4X 1.50 GHz. This low-cost device (~50 euros) is suited
130 to be housed in a small and compact underwater device (Section 2.8). As a minimal capture
131 device, an infrared camera equipped with a 5 megapixel OV5647 sensor and adjustable-
132 focus was targeted (~20 euros), which can be connected to the Pi-4 camera port to se-
133 quentially capture 1080p-resolution images. This camera is endowed with two IR lights
134 activated by photo-resistor light sensors.

135 *2.2. Overall Software Workflow*

136 UDMOS was developed in Java to maximise platform independence and installation
137 flexibility on other hardware than Raspberry Pi-4. The workflow is an endless single-
138 threaded process (Figure 1), with proper internal system-recovery mechanisms (not re-
139 ported in the schema for simplicity). It embeds two types of object detection models -
140 based on a deep learning model (Section 2.3) and an unsupervised model (Section 2.4)
141 respectively - and one movement detection model (Section 2.5) which can be used alter-
142 natively, depending on the computational power of the available hardware.

143 As a first step, the process reads a workflow configuration file that specifies:

- 144 • The input camera frame rate;

- 145 • The Operating System device number of the camera to use;
- 146 • The object/movement detection model to use, i.e. Deep Learning-based, Unsuper-
147 vised, or Standard Movement Detector;
- 148 • Model-specific detection thresholds;
- 149 • Object size classification parameters: The minimum size of the objects to detect
150 (*minSize* parameter); the accepted distance range from the camera (*minDistance* and
151 *maxDistance* parameters); a *sensitivity* parameter for large object classification. All
152 these parameters are explained in Section 2.6;
- 153 • The number of seconds of the video recording session that starts after the detection
154 of a large object (*recordingTime* parameter).

155 It is worth to recall that the *minSize* parameter is the percentage object's size over the
156 camera's frame size that should trigger video recording. The option to adjust the input
157 camera frame rate and device number is meant to manage also alternative input devices
158 (e.g. a USB camera). Alternatively to the device number, the user can specify a range
159 of device numbers with cameras connected. In this case, UDMOS will use all available
160 cameras by processing each frame in a round-robin mode and will record from all cameras
161 when a large object is detected in at least one camera-frame. The entire workflow aims
162 at finding the presence in the scene of candidate objects that may be associated with a
163 fish having size over a certain threshold, where size is indicated as the portion of scene
164 occupied by the object. In this view, it is crucial to estimate the distance of the detected
165 object from the camera, e.g. to exclude small fishes moving close to the camera. Different

166 from other solutions (Harvey et al., 2003; Dunbrack, 2006; Mueller et al., 2006; Schaner
167 et al., 2009; Letessier et al., 2015), this task is approximately solved by UDMOS without
168 additional equipment such as additional cameras or sensors (Section 2.6).

169 The rest of the workflow iteratively captures and processes images from the camera(s).
170 The input image is first downsized (at 640x360 pixels on a Raspberry Pi-4) to reduce cal-
171 culations and power consumption. When more powerful hardware is used, input image
172 resolution can be increased from the configuration file. Based on the first captured im-
173 ages, a classifier estimates the framed scenario as having a *low/medium/high*-brightness
174 depending on the overall illumination of the scene. This classifier uses the following algo-
175 rithm:

Algorithm 1 Brightness scenario classifier

Calculate the geometric mean of the RGB values of the grabbed image

Calculate the log-normal standard deviation of the RGB integer values of the grabbed image

Calculate the confidence limits of the RGB log-normal distribution

For each pixel of the image

If (RGB \leq lower confidence limit) \rightarrow *black pixel*

If (RGB \geq upper confidence limit) \rightarrow *white pixel*

Else \rightarrow *red pixel*

Calculate the ratio of non-black pixels as $B = \frac{\text{white pixels}}{\text{red pixels}}$

If (B \geq 1.5) \rightarrow *high brightness scenario*

If (B \geq 1) \rightarrow *medium brightness scenario*

Else \rightarrow *low brightness scenario*

176 This algorithm is an alternative to statistical binary classification algorithms commonly
177 used for similar tasks (Otsu, 1979; Huang and Wang, 1995; Dominquez et al., 2002). Inter-
178 nally, it uses three classes of pixels (white/black/red) to estimate *how many* outstandingly
179 bright pixels are present in the scene. This operation also distinguishes between deep wa-
180 ter IR light conditions (classified as low-brightness scenarios) and higher brightness condi-
181 tions. Indeed, IR light in deep waters flattens pixel brightness and makes most pixels fall in

182 the *black* cluster. Shallow water scenarios may include highly illuminated spots of sunlight
183 (especially in the morning), alternated with dark zones, and averagely-illuminated zones
184 that may result in an overall high-brightness scenario. In the afternoon, shallow waters
185 would fit medium-brightness conditions. Managing these different underwater conditions
186 required building an *ad hoc* pixel classifier instead of using a standard binary classifier.
187 After brightness classification, scenario-specific values are used for *minSize*, *minDistance*,
188 *maxDistance*, and *sensitivity* from the configuration file. These values were pre-estimated
189 based on a development set of simulated underwater scenarios (Section 2.9).

190 As a further step, the workflow introduces a process to detect either objects or move-
191 ment in front of the camera (Sections 2.3-2.5). Object detectors return a list of rectangles
192 (box-images) inscribing objects in the current frame. This list is the input to the *size*
193 *analyser* process, a decision system that estimates (i) if the inscribed object has a colour
194 distribution likely corresponding to a fish, (ii) the average distance of the object, and (iii)
195 the size of the object. If the object's size is over the *minSize* threshold, the workflow
196 activates a video saving process for *recordingTime* seconds and saves frames into a new
197 MPEG-4 video file. This phase uses a direct Operating System call (through the multi-OS
198 FFmpeg software) to stream frames from the camera to a file, which uses the best cam-
199 era frame rate and image quality. If more than one camera is available, the detection of
200 at least one object with size higher than *minSize* in one camera-frame will trigger video
201 recording from all cameras. After the recording session, the workflow returns back to the
202 camera-frame capturing step.

203 The next sections describe the three detectors currently supported by the UDMOS
204 workflow, i.e. deep learning-based object detector (Section 2.3), unsupervised object de-

205 tector (Section 2.4), and standard movement detector (Section 2.5). Furthermore, the *size*
206 *analyser* is described to explain how object size classification works (Section 2.6). Addi-
207 tionally, the Web service version of UDMOS is explained (Section 2.7), and its embedding
208 equipment for direct underwater operation is described (Section 2.8). Finally, the devel-
209 opment and test sets used to tune and test the workflow are explained (Sections 2.9 and
210 2.10).

211 2.3. *Deep Learning Object Detection Model*

212 As a first object detection model embedded in UDMOS, a state-of-the art deep learn-
213 ing process is used that has the requirement to scale down to low-resource hardware. The
214 You Only Look Once (YOLO) v3 model (Redmon et al., 2016) was chosen because it sup-
215 ports near real-time object detection also on low-resource hardware like Google Coral and
216 NVIDIA Jetson Nano. YOLO is distributed as pre-trained with the Microsoft’s Common
217 Objects in Context (COCO) dataset, a large collection of ~200k annotated images specif-
218 ically conceived for deep learning model training (Lin et al., 2014). It is ~4x faster than
219 alternative models and uses a Convolutional Neural Network (CNN) that divides the image
220 into small cells and classifies every cell as belonging or not to an object (*object-cell* identi-
221 fication). Internally, the CNN uses a decision threshold on the object detection probability,
222 and UDMOS allows to set this threshold in the configuration file (*deep learning* decision
223 threshold parameter). As an additional step, a cell-merging operation estimates bounding
224 boxes around clusters of object-cells (*bounding box* estimation). Finally, each box is la-
225 belled as one among several predefined object classes and is returned as an output. The
226 UDMOS workflow uses YOLO up to the bounding box estimation - i.e. unlabelled boxes
227 are produced as the output - because the pre-trained labelling process was too domain-

228 dependent. Instead, the object detection part of YOLO is independent of the application
229 domain and thus allowed us to re-use a pre-trained YOLO model in UDMOS.

230 However, YOLO is not fast enough to run efficiently on a Raspberry Pi-4, even when
231 using CPU enhancers (Rosebrock, 2020). As an alternative, the Tiny-YOLO version was
232 used, which is 442% faster than YOLO and uses a shallower CNN while losing a small
233 precision percentage. A pre-trained version of Tiny-YOLO was integrated with UDMOS
234 via the DeepLearning4J suite (DL4j, 2016), which efficiently interfaces with the model.
235 Tiny-YOLO requires ~3s to process a camera image at a 416x416 resolution on a Rasp-
236 berry Pi-4. This lag in object detection can be acceptable for slow-dynamic deep water
237 scenarios (below 100m), where fish averagely cover ~1m in 3s (Huse and Ona, 1997;
238 Pinte et al., 2014). However, a more efficient object detector (Section 2.4) is needed to
239 manage faster responses in shallow waters, where speed considerably increases and can
240 even double (Pinte et al., 2014). Using deep learning is more suited for the Web service
241 version of UDMOS that relies on a cloud computing e-Infrastructure (Section 2.7).

242 The deep learning object detection module of UDMOS returns a list of *rectangles*
243 referring to bounding boxes around objects in the current camera frame. The decision
244 threshold regulates the number of produced rectangles. In summary, this module uses the
245 following computational steps (Figure 2-a):

Algorithm 2 deep learning object detection

Adapt the image to the Tiny-YOLO operating resolution (416x416)

Scale the pixel RGB values from the 0-255 range to the 0-1 range

Apply Tiny-YOLO to the image, while using the *deep learning* decision threshold

Retrieve the set of detected objects' bounding boxes (rectangles) from the CNN output

Re-adapt the rectangles' coordinates to fit the original image dimension

Return the set of detected rectangles

246 The image pre-processing steps prepare the frame for the Tiny-YOLO optimal opera-
247 tional resolution, which requires re-adapting the detected rectangles to the original image
248 dimensions afterwards.

249 2.4. *Unsupervised Object Detection Model*

250 The unsupervised object detection model embedded in UDMOS, is a fast object detec-
251 tion method that can run very fast on low-resource hardware. It leverages the illumination
252 properties of typical UDMOS underwater deployment scenarios, where visibility fades out
253 exponentially with distance due to the attenuation characteristics of visible and IR wave-
254 lengths through water. Thus, the background of the scene is darker than the fishes moving
255 in front of the camera and creates a high contrast. This assumption is valid especially in
256 deep waters (e.g., Figure 4-Test case 1), but can be also valid in shallow waters (down

257 to ~50m) if the camera is oriented parallel to the surface, because water rapidly absorbs
258 sunlight (e.g., Figure 4-Test cases 2-5).

259 The unsupervised detection process (Figure 2-b) was entirely realised with Java through
260 the BoofCV library (Abeles, 2017). The contour extraction process is a routine that applies
261 a binary filter to the image using the Otsu thresholding method as a first step (Otsu, 1979).
262 This operation separates pixels into black and white classes by maximising the variance of
263 inter-class intensity. As a second step, 8-neighborhood erosion sets to black those pixels
264 that are not connected to their 8 direct neighbors. The next contour tracing process creates
265 clusters of 8-neighbour continuously connected pixels. A polygonal fit operation is then
266 applied to the contour clusters (BoofCV, 2020). This process first fits a simple polygon of
267 3 sides around the contour line and then increases the number of sides until the euclidean
268 distance between the polygon and the line does not change sensibly. At each computational
269 step, it adds sides by splitting those that likely minimise the polygon-contour distance. As
270 a final process, object bounding boxes are traced by taking the extreme coordinates of each
271 polygon.

272 A configurable *unsupervised model* threshold is introduced to return only boxes of over
273 a minimum size, which controls the relative area of the box $\frac{rectangle\ area}{image\ area}$. This threshold is
274 independent of the scene brightness scenario, because the duty to further select the objects
275 potentially referring to large fishes is left to the *size analyser* (Section 2.6).

276 In summary, the *unsupervised object detection* process uses the following computa-
277 tional steps (Figure 2-b):

Algorithm 3 unsupervised object detection

Transform the image into a binary image with the Otsu method

Apply 8-neighborhood erosion

Trace object contours

Fit closed polygons to contours

For each polygon

 Calculate the bounding box

 Calculate $A = \frac{\text{rectangle area}}{\text{image area}}$

 If ($A \leq \text{unsupervised model threshold}$) \rightarrow discard the rectangle

 Else \rightarrow collect the rectangle

Return the collected rectangles

278 The high contrast between fishes and the background is the main responsible for the
279 effectiveness of this process. Other general unsupervised approaches were experimented
280 as well, e.g. cluster analysis, (Shen et al., 2016) and point of interest detection (Hui and
281 Yuan, 2012), but they gained lower performance because they did not harness operational
282 conditions at best.

283 2.5. Movement Detection Model

284 Standard movement detection is offered as an alternative model to deep learning and
285 unsupervised object detection, and is also a baseline for performance assessment. The
286 movement detection process (schematized in Figure 2-c) is inspired to standard movement
287 detection used in surveillance cameras (Singla, 2014), but adds an adaptive thresholding
288 process that dynamically adjusts detection sensibility. First, the image is divided into three
289 equal zones on the horizontal axis to add directional reference to the detected movements.
290 Second, for each zone, a disparity image is calculated by comparing two consecutive
291 frames pixel-by-pixel. In particular, The relative difference $d_{ij} = \left| \frac{RGB_{current} - RGB_{previous}}{RGB_{previous}} \right|_{ij}$
292 is calculated for every pixel, where ij are the xy coordinates of the pixel in the image. If
293 d is over a *disparity* threshold, the pixel is labeled as *moved*. Based on the disparity im-
294 age, an overall movement score is calculated as the relative number of moved pixels in the
295 zone $M = \frac{Number\ of\ moved\ pixels}{Total\ number\ of\ pixels}$. If M is over a *movement* threshold, a movement detection
296 alert is raised for the reference zone. By construction, the *movement* threshold is a sensi-
297 bility parameter that depends on the scene depth-of-view more than on the brightness of
298 the scene. It regulates the minimum extension of the movement that should trigger video
299 recording. If the *movement* threshold is properly set, a large movement in one zone likely
300 corresponds to a large moving object.

301 The *disparity* threshold is indeed an adaptive threshold that is periodically learned
302 from the average movement of the scene: Every second, the *disparity* threshold is set to
303 $max(d_{ij})$ to cut off repetitive, background, and small movements, and thus to select only
304 large pixel movements within a 1s period. We verified that this approach works also for
305 terrestrial applications, where repetitive movements are present (e.g. trees leaves moved

306 by the wind). The adaptation frequency can be customised from the UDMOS workflow
307 configuration.

308 The *movement detection* process can be summarised as follows (Figure 2-c):

Algorithm 4 Movement detector

Add the input image to a computational stack

If the stack contains less than 2 images → wait

Else, divide the image into three equally spaced left/center/right zones

For each zone

For each pixel ij in the zone

$$\text{Calculate } d_{ij} = \left| \frac{RGB \text{ current} - RGB \text{ previous}}{RGB \text{ previous}} \right|_{ij}$$

If $d_{ij} \geq \text{disparity threshold}$ → label the pixel as *moved*

$$\text{Calculate } M = \frac{\text{Number of moved pixels}}{\text{Total number of pixels}}$$

If $M \geq \text{movement threshold}$ → movement detected in the zone →
 $\text{object found} = TRUE$

If more than 1s has passed from the last update → update the *disparity threshold* to
 $\max(d_{ij})$

If the loop finished without interruption → no moving object was found →
 $\text{object found} = FALSE$

309 One advantage of this algorithm, is that it allows to easily implement a trigger for a

310 motor (e.g. as in the device of Figure 3-b), based on the detection zone, which would move
311 the camera in the direction of the highest movement amount.

312 2.6. *Size Analyser*

313 The *size analyser* is a decision process that analyses every bounding box around the
314 objects detected by one of the supported object detectors. In particular, it (i) decides if
315 the image portion within the box (box-image) potentially refers to a fish, (ii) estimates the
316 distance of the object, and (iii) estimates the object's size. The *size analyser* was designed
317 by balancing speed and accuracy for the limited-resource reference hardware without using
318 distance sensors. It is not used when the movement detector is active, because this process
319 implicitly includes size detection as zone size.

320 As a first step, object bounding boxes overlapping of more than 80% are merged al-
321 together to avoid processing multiple parts of one object. This parameter can be adjusted
322 from the UDMOS configuration. As a second step, each box-image is checked to likely
323 correspond to a single object. Underwater, gray-scaled well-framed objects in front of a
324 camera usually present a colour distribution having a higher brightness in the middle that
325 fades out towards the edges. Middle brightness is higher if a device-mounted light is used
326 (e.g. in deep waters). In order to select well-framed objects only, for each box-image the
327 *size analyser* traces a horizontal histogram of pixel intensity normalised and grouped into
328 5-bins after image gray-scaling. Additionally, it tests this distribution to resemble a 5-bin
329 Gaussian distribution, with unitary standard deviation, through mean squared error estima-
330 tion. If the error is under a *sensitivity* threshold, the object is considered as *well-framed*.
331 This heuristic procedure came after tests on the development set videos, but is also based
332 on the observation that many large fishes targeted by UDMOS (e.g. sharks and tuna) have

333 reflective skins that enhance the described effect. Indeed, non-fish objects in deep waters
334 (e.g. floating algae, plastic debris etc.) usually present non-uniform brightness distribu-
335 tions and thus are discarded by the *size analyser*. Similarly, overlapping fishes typically
336 present multi-modal distributions because they produce several brightness peaks, whose
337 number depends on the number of fishes. Thus, multiple fishes within one bounding box -
338 including fish schools - are normally discarded by this heuristic (Section 3.2).

339 As a consequence, large objects detected by the *size analyser* usually correspond to
340 large fishes in underwater scenarios, especially in deep waters. The choice to select only
341 well-framed objects may limit the recognition of fishes in a perspective position, but these
342 are usually difficult to recognize even for a human expert's eye and even for stereo BRUVs
343 (Cappo et al., 2006; Costa et al., 2006; Ditria et al., 2020). Moreover, selecting only well-
344 framed objects enhances the precision of size estimation.

345 As an additional step, the bounding boxes selected by the brightness analysis are
346 checked to fall within the UDMOS operational distance ranges. To this aim, the box-
347 image distance is estimated as $D = \frac{\text{non-empty bins}}{\text{histogram bins}}$, i.e. as the relative amount of non-black
348 and non-empty bins of the box-image's colour histogram. This idea, validated against
349 development videos, is intuitively valid for underwater scenarios because object colours
350 fade out exponentially with distance. This effect is even more evident with IR light which
351 flattens colours. Thus, the *minDistance* and *maxDistance* configuration parameters define
352 the maximum and minimum percentage of coloured histogram bins that indicate when an
353 object is too close or too far. A very high D percentage corresponds to very close objects,
354 which should be ignored to avoid analysing partial objects and small fishes moving close
355 to the camera. By definition, the *minDistance* value is higher than the *maxDistance* value.

356 As a final step, object size is estimated as $S = \frac{\text{non-black pixels}}{\text{image size}}$, i.e. as the relative number
357 of non-black pixels in the colour histogram of the box-image. The *minSize* configuration
358 parameter is thus the minimum relative size of the object with respect to the entire frame.
359 This estimate is approximately valid because it is calculated for an object that is likely in a
360 longitudinal position and not far from the camera. Thus, it is reasonable to estimate object
361 size as the portion of non black pixels within the box-image.

362 In summary, the *size analyser* process can be summarised as follows:

Algorithm 5 Size analyser

Receive input from the object detection model: a list of rectangular regions in the current frame

Merge rectangles that overlap of more than 80%

For each rectangle

 Extract the inscribed image and use 8-bit gray-scale representation

 Trace the colour histogram

 Trace the horizontal RGB distribution

 Calculate the average error between the horizontal RGB distribution and a 5-bin Gaussian distribution

 If ($\text{error} \leq \text{sensitivity}$) \rightarrow discard the rectangle

 Else calculate $D = \frac{\text{Non-empty bins}}{\text{histogram bins}}$

 If ($D \leq \text{maxDistance}$ OR $D \geq \text{minDistance}$) \rightarrow discard the rectangle

 Else calculate $S = \frac{\text{Non-black pixels}}{\text{image size}}$

 If ($S \leq \text{minSize}$) \rightarrow discard the rectangle

 Else return $\rightarrow \text{large object found} = \text{TRUE}$

If the loop finished without interruption \rightarrow no large object was found \rightarrow
 $\text{large object found} = \text{FALSE}$

363 The output of the process is a boolean variable whose *TRUE* value makes the overall
364 UDMOS workflow activate a camera recording process for *recordingTime* seconds.

365 2.7. Web Service

366 Our workflow is open-source (Supplementary material) and was integrated with the
367 DataMiner cloud computing platform of the D4Science e-Infrastructure (Coro et al., 2017,
368 2015a). This platform allows uploading video files on an online file system and executing
369 the complete UDMOS workflow to retrieve video segments that include large-fish events.
370 DataMiner offers 15 machines with Ubuntu 18.04.5 LTS x86 64 operating system, 16 vir-
371 tual cores, 32 GB of RAM, and 100 GB of disk, to run executions in parallel/distributed
372 and multi-tenancy modes. Moreover, it enables Open Science features like repeatability,
373 reproducibility, re-usability to the integrated processes and enacts collaborative experi-
374 mentation. It also includes an automatic provenance tracking feature, i.e. it keeps track of
375 all the input/output data, parameters, and metadata used (Assante et al., 2019b).

376 UDMOS was integrated through a software-to-service integration tool of DataMiner
377 (Coro et al., 2016c) that published the workflow under the Web Processing Service in-
378 vocation standard (WPS, Schut and Whiteside (2007)), which optimises service re-use
379 from other software. The tool automatically generated a Web graphic interface based on
380 the input/output definitions. Deploying UDMOS as a free-to-use Web service through
381 D4Science also allowed to have low maintenance costs through a long-term sustainability
382 plan of the e-Infrastructure (Assante et al., 2019b).

383 In summary, UDMOS was also published as a distributed, secure, and Open Science
384 Web service. In particular, it is currently available as a WPS service after free registration
385 to the D4Science platform (CNR, 2020). This service accepts a video file as input and

386 asks for confirmation or modification of the default configuration parameters. The ser-
387 vice execution returns one ZIP file containing video clips of large objects captured by the
388 workflow, where each video has *recordingTime* length. Offering UDMOS as a free-to-use
389 Web service on a sustainable platform also allows to go beyond embedded devices and to
390 provide a post-processing system for the videos collected by other BRUVs.

391 2.8. Underwater Deployment Equipment

392 The deployment system comprises of five key components (Figure 3-a): A battery, a
393 data storage device, an embedded single-board *edge* computer capable of running UDMOS
394 (e.g. a Raspberry Pi-4), an IR camera (imaging sensor), and IR LEDs optionally equipped
395 with with photo-resistor light sensors (Figure 3-b). These components are housed inside a
396 waterproof container with an optical window of sufficient size for both the IR emitters to
397 illuminate the study location and the imaging sensor to monitor and capture data (Figure
398 3-c). The imaging sensor differs from visible light cameras in that it lacks an IR filter, thus
399 enabling the capture of IR wavelengths reflected from the study area. The IR emitters have
400 diffuse illumination characteristics in order to illuminate the field of view of the camera
401 sensor evenly without spotlight effect. The quality of the canister housing depends of
402 the operational context of UDMOS and can range from inexpensive micro cases (~ 20
403 euros) for shallow waters and short video sessions (~ 1h), to high-quality solutions for
404 deep waters, e.g. the polyoxymethylene case in Figure 3-c (~ 600 euros) or even more
405 expensive equipment.

406 Data can be captured to either storage integrated within the edge computer or to an ex-
407 ternal storage device such as a solid-state disk drive connected via USB. Solid-state storage
408 also provides benefits in terms of both low power consumption (which is a key attribute of

409 the system in order to maximise monitoring) and capture duration. Additionally, scalable
410 storage offers greater capacity options rather than memory cards and conventional storage
411 limitations which are typical of contemporary BRUV systems.

412 Due to the combination of modifiable characteristics, the system offers a more focussed
413 data capture method and less invasive approach over a longer time period, in comparison
414 to the investment in equipment. The system can be deployed at the study area in the same
415 methods as current systems, either by being physically placed by divers, or deployed from
416 a boat.

417 2.9. *Development Cases*

418 Following best practice indicated by other works (Di Benedetto et al., 2019, 2020),
419 development case videos were built by modelling, rigging, and animating fishes with the
420 Autodesk Maya software (Autodesk, 2010). The aim of testing these videos was to assess
421 the optimal parameters of the detection models and of the *size analyser*. Thus, virtual
422 scenes were produced with both IR and visible light filters, and included virtual groups
423 of fishes of different sizes moving in front of a virtual camera in deep and shallow water
424 conditions. The animations also simulated different speeds of the virtual fishes over time
425 and the presence of multiple fishes at the same time in the camera frame, also having
426 similar sizes to allow the fine tuning of the models.

427 2.10. *Test Cases*

428 The performance of the UDMOS workflow was tested on five videos recorded by un-
429 derwater remote devices in real deployment scenarios under different depth and light con-
430 ditions. The target species to record were tuna, sharks and mantas. All videos had around

431 5 minute and 30 second lengths and included *events* where large sharks, tunas, or rays
432 passed in front of the camera. Events had variable durations with a minimum of ~ 15
433 seconds. Test videos were either self made (the ones in shallow waters) or taken from
434 reusable online material (which limited the number of videos that could be used), and all
435 come from collections of baited and non-baited underwater videos for biodiversity mon-
436 itoring (see the Supplementary material). UDMOS was set to work on 640x320 scaled
437 images to simulate a real operative scenario with a Raspberry Pi-4. The characteristics of
438 the test cases (summarised in Table 1) are reported in the following:

- 439 • **T1** (Figure 4-Test case 1): A set of 15 short videos from deep water environments
440 at depths ranging between ~100m and ~700m, with IR illumination and visibility
441 up to ~1m. A bait was used in most scenarios to attract fishes, and non-fish moving
442 objects were also present in the scene. Target species to record were fishes relatively
443 larger than the others in the scene, e.g. sharks, tunas, and giant squids. The overall
444 video-set duration is 5 min. and 45s.
- 445 • **T2** (Figure 4-Test case 2): A shallow water scenario (~3m depth), with no bait used,
446 and visible light had a ~10m maximum visibility. Density of fishes was generally
447 low but small fishes moving in front of the camera were present. Furthermore, a
448 high sunlight illumination came from the upper-right part of the scene that reduced
449 fish/background contrast. Target species to record were sharks and mantas. The
450 difficulties brought by this scene are a non-uniform illumination and the need to
451 distinguish between large fishes far from the camera and small fishes very close to
452 the camera. Video duration is 5 min. and 16s.

- 453 • **T3** (Figure 4-Test case 3): The same location as T2, with a higher level of illumina-
454 tion. Several shoals were present that UDMOS had to automatically ignore. Video
455 duration is 5 min. and 42s.
- 456 • **T4** (Figure 4-Test case 4): The same location as T2, with a lower level of illumini-
457 nation, many small fishes insistently swimming in front of the camera, and just one
458 event (a shark) to capture. This was the most difficult scenario. Video duration is 5
459 min. and 19s.
- 460 • **T5** (Figure 4-Test case 5): The same location as T2, with a bit higher level of illumini-
461 nation than in T4. Events were more frequent than in T4, but there were more fishes
462 insistently swimming in front of the camera. Video duration is 5 min. and 52s.

463 These descriptions indicate that these five cases were selected to test the limitations
464 and the performance of UDMOS at the variation of fish density, large fishes' distance, and
465 illumination level.

466 2.11. Performance Metrics

467 Performance metrics were defined after dividing the test videos into 15s segments (the
468 minimum duration of an event across the test cases). Segments containing an event were
469 considered *true positives* (TPs), and *true negatives* (TNs) otherwise. A missed 15s event
470 was considered a *false negative* (FN), and a misidentified segment was considered a *false*
471 *positive* (FP). Based on these assumptions, the following standard performance metrics
472 were used:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F\text{-measure} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

473 In summary, *Precision* measures the fraction of events that are truly associated with
474 large fishes passing in front of the camera. *Recall* measures how many correct events are
475 returned among those really occurring. *Accuracy* and *F-measure* summarise the overall
476 quality of the workflow: The former measures how well the workflow is able to detect
477 correct events and to discard incorrect events; the latter is the harmonic mean of Precision
478 and Recall and indicates how much the workflow is balanced between returning many
479 events and correct events.

480 **3. Results**

481 This section reports the UDMOS parameter estimation out of the development set (Sec-
482 tion 3.1), and then reports the workflow performance on our five test cases at the variation
483 of the detection model used (Section 3.2).

484 *3.1. Operational parameter estimation*

485 The tuning of UDMOS on the development set estimated the values of all configura-
486 tion parameters (Table 2): The optimal *deep learning* threshold was set independent of the
487 brightness scenario. Its small value (0.1) makes Tiny-YOLO produce many object bound-
488 ing boxes than using higher values. Likewise, the *unsupervised model* threshold was set
489 independent of the brightness scenario and its value (1%) enabled the selection of even

490 very small bounding boxes. The *movement* threshold depended on the brightness scenario
491 and its values - that refer to the minimum size of the moving pixel blob to detect - were
492 higher for the low brightness scenario (10%), which indicates that large objects correspond
493 to large pixel blobs when the depth of field is shorter. Instead, in the other scenarios large
494 fishes can be also far from the camera, thus the model was set to be more sensitive to
495 movement detection (i.e. *movement* threshold was set to 1%). This threshold is not too
496 weak because it is coupled with the adaptive threshold that excludes small moving blobs
497 such as those associated with shoals and floating algae.

498 The *minSize* threshold was set to decrease when passing from low to high brightness
499 scenarios (15% to 2%) because large fishes correspond to larger bounding boxes in deep
500 water environments and far objects are invisible (visibility is $\sim 1\text{m}$), whereas far objects in
501 the other scenarios may have also small bounding boxes. Also *minDistance* and *maxDis-*
502 *tance* had decreasing trends for the same reason (from 97%-48% to 36%-10%).

503 Finally, the *sensitivity* threshold had an increasing trend across the brightness scenarios
504 (from 3 to 7), since in deep waters objects are close and have a high definition, and thus
505 the Gaussian fit should be stricter to enhance detection Precision. Instead, in the higher
506 brightness scenarios farther large objects can be small and less defined, thus a higher error
507 should be allowed.

508 3.2. *Performance comparison*

509 This section reports a comparison between the UDMOS workflow alternatively us-
510 ing deep learning, unsupervised, and motion detection models. The performance (Table 3)
511 was the same when using a Raspberry Pi-4 and the cloud computing service; thus hardware
512 difference only affected the efficiency of the process but not its effectiveness. The perfor-

513 mance comparison in Table 3 highlights that in the deep water scenarios (T1) all workflows
514 reached at least 75.9% Accuracy and 80% F-measure (Table 3), but particularly the ones
515 based on the deep learning and unsupervised models (100% Accuracy and F-measure).
516 The movement detection workflow reported a 6.7% lower Recall than the other two work-
517 flows because it missed large fishes not moving in front of the camera that were just eating
518 the bait. It also reported a 30% lower Precision because scenes were very animated, and
519 thus recording was often triggered (75.9% Accuracy and 80% F-measure). On T2, the op-
520 timal workflow was the one using the unsupervised object detector (85.7% Accuracy and
521 89.7% F-measure) because it was ~6% more precise at detecting far fishes than the deep
522 learning-based workflow (82.4% Accuracy and 85.7% F-measure). The movement detec-
523 tion workflow had a lower performance than the other two workflows (66.7% Accuracy
524 and 78.8% F-measure) because it also captured small and close moving fishes. On T3, the
525 performance of the unsupervised detector-based workflow was still optimal (91.3% Accu-
526 racy and 93.8% F-measure). Unlike the deep learning-based workflow (76.2% Accuracy
527 and 78.3% F-measure), it ignored shoaling events. Interestingly, the movement detection
528 workflow reached a high performance (80% Accuracy and 87.2% F-measure) because the
529 adaptive threshold correctly classified shoals as small moving blobs. On T4 (the most
530 difficult case), there was a higher heterogeneity between the results due to a different man-
531 agement of small and close fishes. The workflow using the deep learning model had a
532 much better management of false positives (61.1% Accuracy and 58.8% F-measure) than
533 the one using the unsupervised model (38.9% Accuracy and 56% F-measure). The move-
534 ment detector also reached a good performance (47.6% Accuracy and 64.5% F-measure),
535 but in this case a high Recall can be a negative indicator because it suggests that the pro-

536 cess was raising detection alerts too often, although events were rare. In this view, the deep
537 learning-based workflow was the best balance between Precision and Accuracy. Being T5
538 similar to T4, the deep learning-based workflow was better (47.4% Accuracy and 64.3%
539 F-measure) than the one using the unsupervised model (44.4% Accuracy and 58.3% F-
540 measure), and the movement detector reached good performance too (43.5% Accuracy
541 and 60.6% F-measure).

542 Considering the test cases as one overall test set (*Total* row in Table 2), the unsuper-
543 vised detector-based workflow had the highest performance (75.2% Accuracy and 81.9%
544 F-measure), although not much higher than the one of the deep learning-based workflow
545 (74.5% Accuracy and 79% F-measure). The movement detector workflow had a lower per-
546 formance (64.5% Accuracy and 74.9% F-measure) with a ~14% relative Accuracy loss.
547 This result particularly justifies the preference of object detection models over movement
548 detection for the operational scopes of UDMOS. In this difference, the *size analyser*'s role
549 is crucial because the object detectors are both set to detect many objects, and the size
550 analyser is responsible for increasing Precision. Indeed, the detectors report objects also
551 during non-event segments, but these do not become false positives because the *size anal-*
552 *yser* discards them. This feature indicates the added value brought by the *size analyser*
553 to drastically increase the Precision of the underlying models. On the other hand, also
554 the Precision of the object detectors is important. For example, the higher precision of
555 Tiny-YOLO is the main responsible for the different results on T4, because the two object
556 detectors use the same *size analyser*.

557 Apart from effectiveness measurements, the efficiency of the models is also important
558 because UDMOS is meant to run on a low-resource device like Raspberry Pi-4. When

559 using the deep learning model on this platform, UDMOS takes ~3s to process one frame,
560 which makes it unpractical for fast-moving fish detection. Instead, UDMOS takes ~800ms
561 per frame when using the unsupervised model, and ~400ms per frame with the movement
562 detector. Thus, the unsupervised model is a good compromise between efficiency and ef-
563 fectiveness when using low-resource hardware. Power consumption is also well managed
564 thanks to the single-threaded design of the workflow and an accurate internal management
565 of memory. A Raspberry Pi-4 can continuously compute UDMOS with the unsupervised
566 model for ~4h with a 5000mAh/5V standard power bank, which guarantees a much longer
567 duration with professional power supply (~66000 mAh/22.2V).

568 **4. Discussion and Conclusions**

569 In this paper, a workflow to detect large objects underwater, with the objective of de-
570 tecting large fishes moving in front of a camera, has been described. The presented so-
571 lution has a number of advantages (Table 4), e.g. it can internally use one among three
572 different models that reported reasonably good performance on the selected test cases. The
573 results also demonstrate that UDMOS can work effectively and efficiently even with low-
574 resource hardware. Deep water scenarios are particularly suited to our approach because
575 fish movements are relatively slow and IR light and limited visibility enhance distance
576 and object size estimation. Generally, the time lag of an object detection algorithm is
577 appropriate for certain operational conditions when the average distance covered by the
578 target species during the lag is lower than the extent of the camera frame, otherwise a
579 fish could cross the camera during the lag without being seen. For example, the 3s lag
580 of our deep learning-based workflow is suited for targeting sharks and rays at a ~200m

581 depth because these species averagely cover 1m in 3s (Pinte et al., 2014), which corre-
582 sponds to ~64% of the camera frame at a 1.5m distance from an OV5647 sensor (i.e.,
583 $\frac{1m * focal-length}{1.5m * sensor-image-width} = 0.64$, according to the specifications of the Raspberry Pi Founda-
584 tion (2021)). These considerations should be evaluated when selecting the most appropri-
585 ate workflow for the target species and operational conditions.

586 Our approach can implement a low-cost solution to (i) monitor fish size, (ii) enable
587 large underwater monitoring networks with contained costs to be deployed, and (iii) mon-
588 itor the presence of indicator species or behavioural traits over a large area for a long time.
589 Cost effectiveness should be also considered in the light to spare expert analysts' time.
590 For example, the presented test case 4 contained just one event of a shark passing (for ~2
591 minutes) far from the camera and included irrelevant events for 62% of video time. For
592 longer videos this percentage can be much higher. In this case, the deep learning-based
593 workflow reported 1 minute and 15s of the event and 30s of other irrelevant events (71.4%
594 Precision). Thus, it was able to detect the presence of the targeted shark and produced
595 only two irrelevant videos of 15s, i.e. just a 5% over the total video time.

596 The high Precision on event identification across all test cases indicates that UDMOS
597 was accurate at measuring the minimum target fish-length, which is enough for the aim of
598 our workflow to aid human experts and automatic systems to identify species and estimate
599 abundance and biomass. Indeed, the captured videos have an overall length that is aver-
600 agely much lower than a continuous recording and contain well-framed species facilitating
601 post-processing tasks. Adding further processing on-board - e.g., upper size limitation and
602 species identification - would have (i) introduced unnecessary bias, (ii) reduced event cap-
603 turing Precision, and (iii) consumed more power.

604 UDMOS has also scalability features that automatically support the improvement of
605 the quality of both camera and processing hardware, while keeping power consumption
606 low. It is worth to note that using a higher-resolution camera would be beneficial more to
607 the human observers (e.g. to recognize the detected species) than to the object detection
608 models, which work well also with low image resolutions (Redmon and Farhadi, 2018).
609 The use of Java as a programming language aims at covering other single-board comput-
610 ers and ARM-CPU based platforms, but our solution can work directly also on powerful
611 computers. Differently from other systems (Letessier et al., 2015; González-Rivero et al.,
612 2020; Hermann et al., 2020), costs are also reduced in the multiple operational contexts
613 covered by our solution without model re-training.

614 Apart from underwater devices, UDMOS is also offered as a free-to-use Open Science
615 Web service. In this version, UDMOS could be automatically invoked on video streams
616 to extract the presence of large fishes in a certain area. This service has the same effec-
617 tiveness of the on-board process if the workflow is suited to the application case, because
618 UDMOS addresses the easier goal to identify a minimum fish length instead of a specific
619 fish length or species. The availability of Open Science features to reproduce results and
620 trace computational provenance, guarantees the transparency of the produced results to-
621 wards stakeholders and enables inter-scientist collaboration through the sharing of input,
622 output, and parameters.

623 UDMOS could be used to implement semi-automatic batch monitoring analyses for
624 large fish presence in a certain area. For example, to monitor the presence of elasm-
625 branches at a specific location such as a sea mount or monitor anthropogenic relationships in
626 the context of Other Effective Area-Based Conservation Measures (OECM, CBD (2018)).

627 UDMOS could also help stakeholders to estimate the average size of sharks across longer
628 time periods. Furthermore, big data processing methodologies suggest that viable infor-
629 mation can be extracted even from noisy temporal observations of large fish occurrences
630 (Froese et al., 2014). For example, features like seasonal species composition, average
631 overall fish size change in time, and risk indicators could be automatically inferred with a
632 good reliability (Coro et al., 2016, 2018).

633 Finally, although UDMOS was conceived to optimally operate in underwater scenar-
634 ios, it embeds a movement detection model that allows to extend its application contexts.
635 Indeed, this model showed a reasonably high performance on several underwater test cases
636 and is 2 times faster than the unsupervised object detector. Furthermore, it adds directional
637 information that could be used by motors connected to the computational device. Thus,
638 this process makes UDMOS potentially usable also in terrestrial applications, where dy-
639 namics are averagely faster and automatic movement detection is effective.

640 **Acknowledgments**

641 This work was conducted under the self-funded ISTI CNR-Visual Persistence collab-
642 oration agreement Number ISTI-0020363/2020. Gianpaolo Coro acknowledges the cour-
643 tesy of Dr. Edith Widder and Dr. Nathan Robinson (NOAA OER) for providing a video
644 of a rare giant squid *Architeuthis dux* captured at a 700m depth by a baited underwater
645 device, which was included in Test Case 1. The authors also acknowledge the courtesy of
646 Alexander Wilson (University of Plymouth) to allow using a video footage on isopods and
647 giant squids in Test Case 1 (Wilson et al., 2017). Visual Persistence was supported with
648 funding from the University of Exeter, Exeter Marine Research Group.

649 **References**

- 650 Abeles, P., 2017. Boofcv project website. <https://boofcv.org>.
- 651 Apps, P.J., McNutt, J.W., 2018. How camera traps work and how to work them. *African*
652 *Journal of Ecology* 56, 702–709.
- 653 Assante, M., Candela, L., Castelli, D., Cirillo, R., Coro, G., Frosini, L., Lelii, L., Man-
654 giacrapa, F., Pagano, P., Panichi, G., 2019b. Enacting open science by d4science. *Future*
655 *Generation Computer Systems* 101, 555–563.
- 656 Autodesk, 2010. Autodesk Maya. Accessed on Dec. 2020 [https://www.autodesk.](https://www.autodesk.com/products/maya/overview)
657 [com/products/maya/overview](https://www.autodesk.com/products/maya/overview).
- 658 BoofCV, 2020. Polyline split and merge process documentation. Ac-
659 cessed Dec. 2020 [http://boofcv.org/javadoc/boofcv/alg/shapes/](http://boofcv.org/javadoc/boofcv/alg/shapes/polyline/splitmerge/PolylineSplitMerge.html)
660 [polyline/splitmerge/PolylineSplitMerge.html](http://boofcv.org/javadoc/boofcv/alg/shapes/polyline/splitmerge/PolylineSplitMerge.html).
- 661 Brooks, E.J., Sloman, K.A., Sims, D.W., Danylchuk, A.J., 2011. Validating the use of
662 baited remote underwater video surveys for assessing the diversity, distribution and
663 abundance of sharks in the bahamas. *Endangered Species Research* 13, 231–243.
- 664 Candela, L., Castelli, D., Coro, G., Pagano, P., Sinibaldi, F., 2016. Species distribution
665 modeling in the cloud. *Concurrency and Computation: Practice and Experience* 28,
666 1056–1079.
- 667 Cappo, M., Harvey, E., Shortis, M., 2006. Counting and measuring fish with baited video
668 techniques-an overview, in: *Australian Society for Fish Biology Workshop Proceedings,*
669 *Australian Society for Fish Biology Tasmania.* pp. 101–114.

670 Cappelletti, M., Speare, P., De'ath, G., 2004. Comparison of baited remote underwater video
671 stations (bruv) and prawn (shrimp) trawls for assessments of fish biodiversity in inter-
672 reefal areas of the great barrier reef marine park. *Journal of Experimental Marine Biol-
673 ogy and Ecology* 302, 123–152.

674 CBD, 2018. Decision adopted by the Conference of the Parties to the Convention on
675 Biological Diversity. 14/8. Protected areas and other effective area-based conserva-
676 tion measures. Conf. Parties to Conv. Biol. Divers. Fourteenth Meet. Agenda item
677 24, CBD/COP/DEC/14/8 30 November 2018, 1–19. Accessed Mar. 2021 <https://www.cbd.int/doc/decisions/cop-14/cop-14-dec-08-en.pdf>.
678

679 CNR, 2020. Underwater Detector of Moving Object Size as-a-service. [https://services.d4science.org/group/rprototypinglab/data-miner?
680 OperatorId=org.gcube.dataanalysis.wps.statisticalmanager.
681 synchserver.mappedclasses.transducerers.UDMOS](https://services.d4science.org/group/rprototypinglab/data-miner?OperatorId=org.gcube.dataanalysis.wps.statisticalmanager.synchserver.mappedclasses.transducerers.UDMOS).
682

683 Codd-Downey, R., Jenkin, M., Allison, K., 2017. Milton: An open hardware underwa-
684 ter autonomous vehicle, in: 2017 IEEE International Conference on Information and
685 Automation (ICIA), IEEE. pp. 30–34.

686 Coghlan, A., McLean, D., Harvey, E., Langlois, T., 2017. Does fish behaviour bias abun-
687 dance and length information collected by baited underwater video? *Journal of Experi-
688 mental Marine Biology and Ecology* 497, 143–151.

689 Connolly, R., Fairclough, D., Jinks, E., Ditria, E., Jackson, G., Lopez-Marcano, S., Olds,
690 A., Jinks, K., 2021. Improved accuracy for automated counting of a fish in baited
691 underwater videos for stock assessment. *bioRxiv* .

- 692 Coro, G., Candela, L., Pagano, P., Italiano, A., Liccardo, L., 2015a. Parallelizing the
693 execution of native data mining algorithms for computational biology. *Concurrency
694 and Computation: Practice and Experience* 27, 4630–4644.
- 695 Coro, G., Large, S., Magliozzi, C., Pagano, P., 2016. Analysing and forecasting fisheries
696 time series: purse seine in indian ocean as a case study. *ICES Journal of Marine Science*
697 73, 2552–2571.
- 698 Coro, G., Panichi, G., Pagano, P., 2016c. A web application to publish r scripts as-a-
699 service on a cloud computing platform. *Bollettino di Geofisica Teorica ed Applicata* 57,
700 51–53.
- 701 Coro, G., Panichi, G., Scarponi, P., Pagano, P., 2017. Cloud computing in a distributed
702 e-infrastructure using the web processing service standard. *Concurrency and Computa-
703 tion: Practice and Experience* 29, e4219.
- 704 Coro, G., Vilas, L.G., Magliozzi, C., Ellenbroek, A., Scarponi, P., Pagano, P., 2018. Fore-
705 casting the ongoing invasion of *lagocephalus sceleratus* in the mediterranean sea. *Eco-
706 logical Modelling* 371, 37–49.
- 707 Costa, C., Loy, A., Cataudella, S., Davis, D., Scardi, M., 2006. Extracting fish size using
708 dual underwater cameras. *Aquacultural Engineering* 35, 218–227.
- 709 Daum, D.W., 2005. Monitoring fish wheel catch using event-triggered video technology.
710 *North American Journal of Fisheries Management* 25, 322–328.
- 711 Dawkins, M., Sherrill, L., Fieldhouse, K., Hoogs, A., Richards, B., Zhang, D., Prasad, L.,
712 Williams, K., Lauffenburger, N., Wang, G., 2017. An open-source platform for under-

713 water image and video analytics, in: 2017 IEEE Winter Conference on Applications of
714 Computer Vision (WACV), IEEE. pp. 898–906.

715 Di Benedetto, M., Carrara, F., Meloni, E., Amato, G., Falchi, F., Gennaro, C., 2020. Learn-
716 ing accurate personal protective equipment detection from virtual worlds. *Multimedia*
717 *Tools and Applications* , 1–13.

718 Di Benedetto, M., Meloni, E., Amato, G., Falchi, F., Gennaro, C., 2019. Learning
719 safety equipment detection using virtual worlds, in: 2019 International Conference on
720 Content-Based Multimedia Indexing (CBMI), IEEE. pp. 1–6.

721 Ditria, E.M., Lopez-Marcano, S., Sievers, M., Jinks, E.L., Brown, C.J., Connolly, R.M.,
722 2020. Automating the analysis of fish abundance using object detection: optimizing
723 animal ecology with deep learning. *Frontiers in Marine Science* 7, 429.

724 DL4j, 2016. Deeplearning4j: Open-source distributed deep learning for the JVM, apache
725 software foundation license 2.0.

726 Dominquez, J.A., Klinko, S., Voska, N., 2002. Binarization of gray-scaled digital images
727 via fuzzy reasoning. <https://ntrs.nasa.gov/citations/20020094350>,
728 accessed Dec. 2020.

729 Dunbrack, R., 2006. In situ measurement of fish body length using perspective-based
730 remote stereo-video. *Fisheries Research* 82, 327–331.

731 Dunlop, K.M., Marian Scott, E., Parsons, D., Bailey, D.M., 2015. Do agonistic behaviours
732 bias baited remote underwater video surveys of fish? *Marine ecology* 36, 810–818.

- 733 Ebner, B.C., Starrs, D., Morgan, D.L., Fulton, C.J., Donaldson, J.A., Doody, J.S., Cousins,
734 S., Kennard, M., Butler, G., Tonkin, Z., et al., 2014. Emergence of field-based underwa-
735 ter video for understanding the ecology of freshwater fishes and crustaceans in australia.
736 *Journal of the Royal Society of Western Australia* 97, 287–296.
- 737 Edgington, D.R., Cline, D.E., Davis, D., Kerkez, I., Mariette, J., 2006. Detecting, tracking
738 and classifying animals in underwater video, in: *OCEANS 2006, IEEE*. pp. 1–5.
- 739 Edgington, D.R., Salamy, K.A., Risi, M., Sherlock, R., Walther, D., Koch, C., 2003. Au-
740 tomated event detection in underwater video, in: *Oceans 2003. Celebrating the Past.*
741 *Teaming Toward the Future (IEEE Cat. No. 03CH37492), IEEE*. pp. P2749–P2753.
- 742 Froese, R., Thorson, J.T., Reyes Jr, R., 2014. A bayesian approach for estimating length-
743 weight relationships in fishes. *Journal of Applied Ichthyology* 30, 78–85.
- 744 Golkarnarenji, G., Kouzani, A.Z., Semianiw, N.I., Goodall, D., Gilbert, D., Driscoll, D.,
745 2018. Automatic detection of moving baw baw frogs in camera trap videos, in: *2018*
746 *IEEE International Conference on Mechatronics and Automation (ICMA), IEEE*. pp.
747 1112–1116.
- 748 González-Rivero, M., Beijbom, O., Rodriguez-Ramirez, A., Bryant, D.E., Ganase, A.,
749 Gonzalez-Marrero, Y., Herrera-Reveles, A., Kennedy, E.V., Kim, C.J., Lopez-Marcano,
750 S., et al., 2020. Monitoring of coral reefs using artificial intelligence: A feasible and
751 cost-effective approach. *Remote Sensing* 12, 489.
- 752 Harvey, E., Cappo, M., Shortis, M., Robson, S., Buchanan, J., Speare, P., 2003. The accu-
753 racy and precision of underwater measurements of length and maximum body depth of

754 southern bluefin tuna (*thunnus maccoyii*) with a stereo–video camera system. *Fisheries*
755 *Research* 63, 315–326.

756 Harvey, E.S., Cappel, M., Butler, J.J., Hall, N., Kendrick, G.A., 2007. Bait attraction
757 affects the performance of remote underwater video stations in assessment of demersal
758 fish community structure. *Marine Ecology Progress Series* 350, 245–254.

759 Hermann, A., Chladek, J., Stepputtis, D., 2020. ifo (infrared fish observation)–an open
760 source low-cost infrared underwater video system. *HardwareX* 8, e00149.

761 Hsiao, Y.H., Chen, C.C., Lin, S.I., Lin, F.P., 2014. Real-world underwater fish recognition
762 and identification, using sparse representation. *Ecological informatics* 23, 13–21.

763 Huang, L.K., Wang, M.J.J., 1995. Image thresholding by minimizing the measures of
764 fuzziness. *Pattern recognition* 28, 41–51.

765 Hui, D., Yuan, H.D., 2012. Research of image matching algorithm based on surf features,
766 in: *2012 International Conference on Computer Science and Information Processing*
767 *(CSIP), IEEE*. pp. 1140–1143.

768 Huse, I., Ona, E., 1997. Tilt angle distribution and swimming speed of overwintering
769 norwegian spring spawning herring. *Oceanographic Literature Review* 5, 524.

770 Hussey, N.E., Kessel, S.T., Aarestrup, K., Cooke, S.J., Cowley, P.D., Fisk, A.T., Harcourt,
771 R.G., Holland, K.N., Iverson, S.J., Kocik, J.F., et al., 2015. Aquatic animal telemetry: a
772 panoramic window into the underwater world. *Science* 348.

773 Kays, R., Tilak, S., Kranstauber, B., Jansen, P.A., Carbone, C., Rowcliffe, M.J., Fountain,
774 T., Eggert, J., He, Z., 2010. Monitoring wild animal communities with arrays of motion
775 sensitive camera traps. arXiv preprint arXiv:1009.5718 .

776 Knausgård, K.M., Wiklund, A., Sjørdalen, T.K., Halvorsen, K., Kleiven, A.R., Jiao, L.,
777 Goodwin, M., 2020. Temperate fish detection and classification: a deep learning based
778 approach. arXiv preprint arXiv:2005.07518 .

779 Konovalov, D.A., Saleh, A., Bradley, M., Sankupellay, M., Marini, S., Sheaves, M., 2019.
780 Underwater fish detection with weak multi-domain supervision, in: 2019 International
781 Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.

782 Letessier, T.B., Juhel, J.B., Vigliola, L., Meeuwig, J.J., 2015. Low-cost small action cam-
783 eras in stereo generates accurate underwater measurements of fish. *Journal of Experi-*
784 *mental Marine Biology and Ecology* 466, 120–126.

785 Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick,
786 C.L., 2014. Microsoft coco: Common objects in context, in: European conference on
787 computer vision, Springer. pp. 740–755.

788 Mallet, D., Pelletier, D., 2014. Underwater video techniques for observing coastal marine
789 biodiversity: a review of sixty years of publications (1952–2012). *Fisheries Research*
790 154, 44–62.

791 Marcot, B.G., Lorenz, T.J., Fischer, P., Weinstein, B.G., Cowell, S., 2019. Efficacy of
792 automated detection of motion in wildlife monitoring videos. *Wildlife Society Bulletin*
793 43, 726–736.

- 794 Marini, S., Fanelli, E., Sbragaglia, V., Azzurro, E., Fernandez, J.D.R., Aguzzi, J., 2018.
795 Tracking fish abundance by underwater image recognition. *Scientific reports* 8, 1–12.
- 796 Miguel, A., Beery, S., Flores, E., Klemesrud, L., Bayrakkismith, R., 2016. Finding areas
797 of motion in camera trap images, in: 2016 IEEE international conference on image
798 processing (ICIP), IEEE. pp. 1334–1338.
- 799 Mueller, R.P., Brown, R.S., Hop, H., Moulton, L., 2006. Video and acoustic camera
800 techniques for studying fish under ice: a review and comparison. *Reviews in Fish*
801 *Biology and Fisheries* 16, 213–226.
- 802 Otsu, N., 1979. A threshold selection method from gray-level histograms. *IEEE transac-*
803 *tions on systems, man, and cybernetics* 9, 62–66.
- 804 Palazzo, S., Spampinato, C., Giordano, D., 2014. Large scale data processing in ecology:
805 a case study on long-term underwater video monitoring, in: 2014 22nd Euromicro In-
806 ternational Conference on Parallel, Distributed, and Network-Based Processing, IEEE.
807 pp. 312–316.
- 808 Pinte, N., Mallefet, J., Claes, J., 2014. Swimming speed of deep-water sharks inferred
809 from video footage analysis from deep baited camera, in: Proceedings of the Benelux
810 zoology congress, Royal Belgian Zoological Society. p. 1.
- 811 Qin, H., Li, X., Liang, J., Peng, Y., Zhang, C., 2016. Deepfish: Accurate underwater live
812 fish recognition with a deep architecture. *Neurocomputing* 187, 49–58.
- 813 Quevedo, E., Delory, E., Callicó, G., Tobajas, F., Sarmiento, R., 2017. Underwater video

814 enhancement using multi-camera super-resolution. *Optics communications* 404, 94–
815 102.

816 Raspberry Pi Foundation, 2021. Raspberry Pi Camera Modules Specifications. Accessed
817 Mar. 2021 [https://www.raspberrypi.org/documentation/hardware/
818 camera/](https://www.raspberrypi.org/documentation/hardware/camera/).

819 Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified,
820 real-time object detection, in: *Proceedings of the IEEE conference on computer vision
821 and pattern recognition*, pp. 779–788.

822 Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint*
823 [arXiv:1804.02767](https://arxiv.org/abs/1804.02767) .

824 Rosebrock, A., 2020. Yolo and tiny-yolo object detection on the raspberry pi
825 and movidius ncs. [https://www.pyimagesearch.com/2020/01/27/
826 yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-n](https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-n)

827 Salman, A., Siddiqui, S.A., Shafait, F., Mian, A., Shortis, M.R., Khurshid, K., Ulges, A.,
828 Schwanecke, U., 2020. Automatic fish detection in underwater videos by a deep neural
829 network-based hybrid motion learning system. *ICES Journal of Marine Science* 77,
830 1295–1307.

831 Santana-Garcon, J., Braccini, M., Langlois, T.J., Newman, S.J., McAuley, R.B., Harvey,
832 E.S., 2014. Calibration of pelagic stereo-bruv s and scientific longline surveys for sam-
833 pling sharks. *Methods in Ecology and Evolution* 5, 824–833.

834 Schaner, T., Fox, M.G., Taraborelli, A.C., 2009. An inexpensive system for underwater
835 video surveys of demersal fishes. *Journal of Great Lakes Research* 35, 317–319.

836 Schlining, B., Stout, N.J., 2006. Mbari’s video annotation and reference system, in:
837 OCEANS 2006, IEEE. pp. 1–5.

838 Schmid, K., Reis-Filho, J.A., Harvey, E., Giarrizzo, T., 2017. Baited remote underwater
839 video as a promising nondestructive tool to assess fish assemblages in clearwater ama-
840 zonian rivers: testing the effect of bait and habitat type. *Hydrobiologia* 784, 93–109.

841 Schut, P., Whiteside, A., 2007. OpenGIS Web Processing Service. Open Geospatial Con-
842 sortium. Open Geospatial Consortium Technical Reports .

843 Shafait, F., Mian, A., Shortis, M., Ghanem, B., Culverhouse, P.F., Edgington, D., Cline,
844 D., Ravanbakhsh, M., Seager, J., Harvey, E.S., 2016. Fish identification from videos
845 captured in uncontrolled underwater environments. *ICES Journal of Marine Science*
846 73, 2737–2746.

847 Sheaves, M., Bradley, M., Herrera, C., Mattone, C., Lennard, C., Sheaves, J., Konovalov,
848 D.A., 2020. Optimizing video sampling for juvenile fish surveys: Using deep learn-
849 ing and evaluation of assumptions to produce critical fisheries parameters. *Fish and*
850 *Fisheries* 21, 1259–1276.

851 Sheehan, E.V., Bridger, D., Nancollas, S.J., Pittman, S.J., 2020. Pelagicam: A novel
852 underwater imaging system with computer vision for semi-automated monitoring of
853 mobile marine fauna at offshore structures. *Environmental monitoring and assessment*
854 192, 1–13.

- 855 Shen, J., Hao, X., Liang, Z., Liu, Y., Wang, W., Shao, L., 2016. Real-time superpixel
856 segmentation by dbscan clustering algorithm. *IEEE transactions on image processing*
857 25, 5933–5942.
- 858 Singla, N., 2014. Motion detection based on frame difference method. *International*
859 *Journal of Information & Computation Technology* 4, 1559–1565.
- 860 Spampinato, C., Chen-Burger, Y.H., Nadarajan, G., Fisher, R.B., 2008. Detecting, tracking
861 and counting fish in low quality unconstrained underwater videos. *VISAPP (2) 2008*, 1.
- 862 Struthers, D.P., Danylchuk, A.J., Wilson, A.D., Cooke, S.J., 2015. Action cameras: bring-
863 ing aquatic and fisheries research into view. *Fisheries* 40, 502–512.
- 864 Van Damme, T., 2015. Computer vision photogrammetry for underwater archaeological
865 site recording in a low-visibility environment. *International Archives of the Photogram-*
866 *metry, Remote Sensing & Spatial Information Sciences* .
- 867 Vos, L.D., Götz, A., Winker, H., Attwood, C., 2014. Optimal bruvs (baited remote under-
868 water video system) survey design for reef fish monitoring in the stilbaai marine pro-
869 tected area. *African Journal of Marine Science* 36, 1–10. doi:10.2989/1814232X.
870 2013.873739.
- 871 Wilson, A.D., Szekeres, P., Violich, M., Gutowsky, L.F., Eliason, E.J., Cooke, S.J., 2017.
872 Activity syndromes and metabolism in giant deep-sea isopods. *Deep Sea Research*
873 *Part I: Oceanographic Research Papers* 121, 237 – 244. URL: <http://www.sciencedirect.com/science/article/pii/S0967063716303739>,
874 doi:<https://doi.org/10.1016/j.dsr.2017.02.003>.

876 Yoon, S., Azad, A.K., Oh, H., Kim, S., 2012. Aurrp: An auv-aided underwater routing
877 protocol for underwater acoustic sensor networks. *Sensors* 12, 1827–1845.

878 Zhou, H., Kimber, D., Turner, A., 2008. System and method for process segmentation
879 using motion detection. US Patent App. 11/504,277.

	Test cases				
	T1	T2	T3	T4	T5
Number of events	15	3	3	1	4
Number of events (15s chunks)	15	9	13	10	10
Video duration (s)	345	316	342	319	352
Depth (m)	100-700	3	3	3	3
Field of view (m)	1	10	10	10	10
Light	Infrared	Visible	Visible	Visible	Visible
Brightness scenario	Low	High	High	Medium	High
Fish density	High/Low	Low	Medium	Low	Low
Bait	Yes/No	No	No	No	No
Main features tested	Management of low visibility and IR light	Far fish detection in adverse illumination conditions	Shoal discard	Management of small and close fishes and rare events	Management of many small and close fishes

Table 1: Summary of the characteristics of the five test videos used to evaluate our methodology. The 15 deep-sea test videos used in T1 referred to scenarios with depths ranging between 100m and 700m and included both baited and unbaited devices, mostly with few fishes in the scenes and sometimes with shoals present. Test cases T2-T4 come from shallow water (10m) unbaited systems.

Threshold name	Brightness scenario		
	Low	Medium	High
deep learning	0.1	0.1	0.1
unsupervised model	1%	1%	1%
movement	10%	1%	1%
minSize	15%	3%	2%
minDistance	97%	45%	36%
maxDistance	48%	10%	10%
sensitivity	3	6	7

Table 2: Workflow thresholds estimated based on our development-set videos. The deep learning and unsupervised model thresholds are the only parameters independent of the brightness scenario.

Deep learning-based object detector				
	Precision	Recall	Accuracy	F-Measure
T1	100%	100%	100%	100%
T2	75.0%	100.0%	82.4%	85.7%
T3	90.0%	69.2%	76.2%	78.3%
T4	71.4%	50.0%	61.1%	58.8%
T5	50.0%	90.0%	47.4%	64.3%
Total	75.8%	82.5%	74.5%	79.0%
Unsupervised object detector				
T1	100%	100%	100%	100%
T2	81.3%	100.0%	85.7%	89.7%
T3	88.2%	100.0%	91.3%	93.8%
T4	46.7%	70.0%	38.9%	56.0%
T5	50.0%	70.0%	44.4%	58.3%
Total	74.7%	90.8%	75.2%	81.9%
Movement detector				
T1	70.0%	93.3%	75.9%	80.0%
T2	65.0%	100.0%	66.7%	78.8%
T3	77.3%	100.0%	80.0%	87.2%
T4	47.6%	100.0%	47.6%	64.5%
T5	43.5%	100.0%	43.5%	60.6%
Total	60.4%	98.5%	64.5%	74.9%

Table 3: Performance comparison of the presented large-fish detection workflows across all use cases at the change of the detection models used internally.

Advantages	
Support of multiple detection algorithms	The user can choose among three detection algorithms: Deep Learning-based, Unsupervised, or Standard Movement Detector.
Computational hardware scalability	Supported hardware ranges from a very inexpensive Raspberry Pi-4 (or another single-board computer) with a single IR camera, to powerful hardware with multiple cameras. Furthermore, UDMOS is Java-based and thus supports multiple operating systems.
Camera scalability	Works automatically with one to N cameras connected to the computing platform.
Multiple illumination operational conditions	Works with both IR and visible light because it automatically selects its optimal configuration based on the scene brightness.
Management of common issues on large fish detection found by other systems	Uses a statistical analysis on colour distribution to account for too close fishes, overlapping fishes, and non-fish objects.
Support of both edge- and cloud-computing	Offered both as an embeddable solution for direct underwater operations and as a Web service for batch video processing.
Compatibility with species identification systems	The produced videos can be used directly to feed automatic species identification systems.
Cost effectiveness of post-processing	Reduces human expert's time and automatic processing time for analysing videos.
Low power consumption	The single-threaded architecture is able to spare power consumption and achieve ~1 week of continuous activity with professional deployment equipment.
Open Science compliance	The Open Science compliance of the Web service makes it easy to invoke the service for video-stream processing and guarantees the transparency of the results through open repeatability, reproducibility, and provenance tracking.
Potential limitations	
Processing time lag	The deep learning-based workflow has single-frame processing time that can be unsuited for low-hardware platforms, and for scenarios with fast movements to be captured.
No upper-bound for fish size detection	There is no upper size bound of the fishes to detect, in order to maximise detection effectiveness.
No on-board species identification	On-board species identification is not supported due to target-hardware limitations.
Motion detection is affected by attracted fishes	The motion detection-based workflow is very sensitive to small fishes moving in front of the camera.
Sensitivity to non-uniform illumination	Distance estimation is based on colour fading and thus non-uniform illumination (especially in shallow waters) can affect event capturing precision.
Sensitivity to uniform distant shoals	Single-species detection uses colour distribution analysis, thus compact distant shoals may present a uniform colour distribution and may be detected as one large fish.

Table 4: Highlight of the advantages and potential limitations of UDMOS.

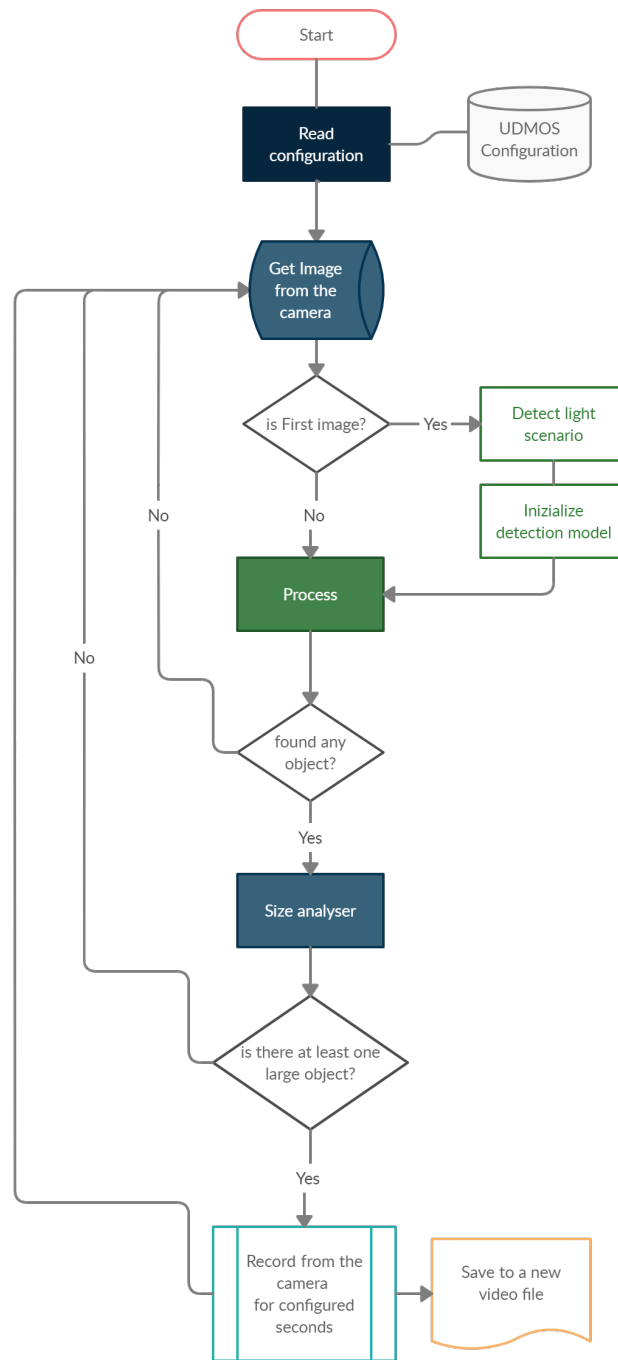


Figure 1: Flowchart of our workflow.

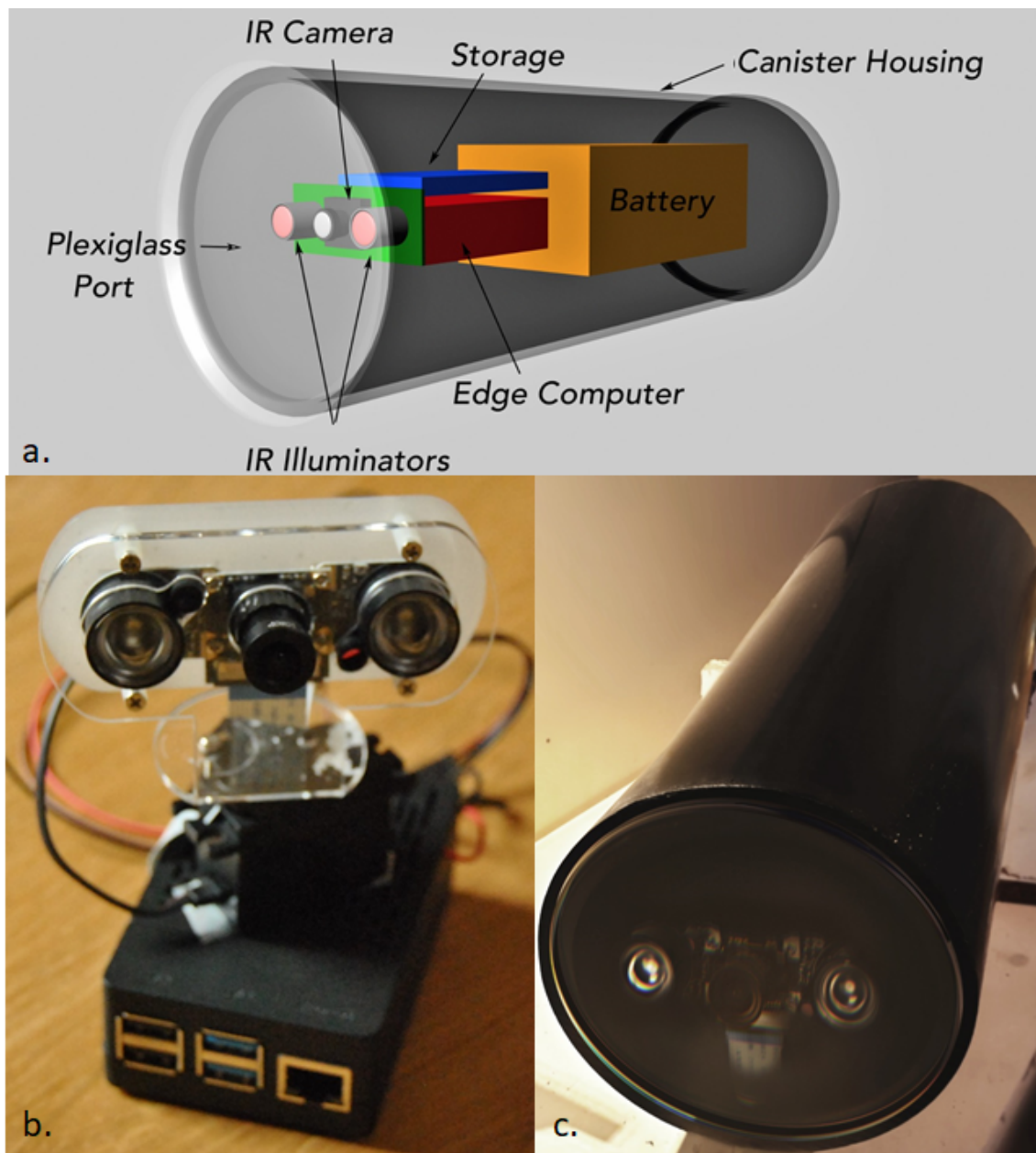


Figure 3: Images of the proposed equipment for remote underwater video operations: a) overall schema of the underwater device; b) Raspberry Pi-4 with IR camera and two IR lights activated by photo-resistor light sensors; c) canister of 160 mm length in polyoxymethylene with IR camera and Raspberry Pi-4 mounted.

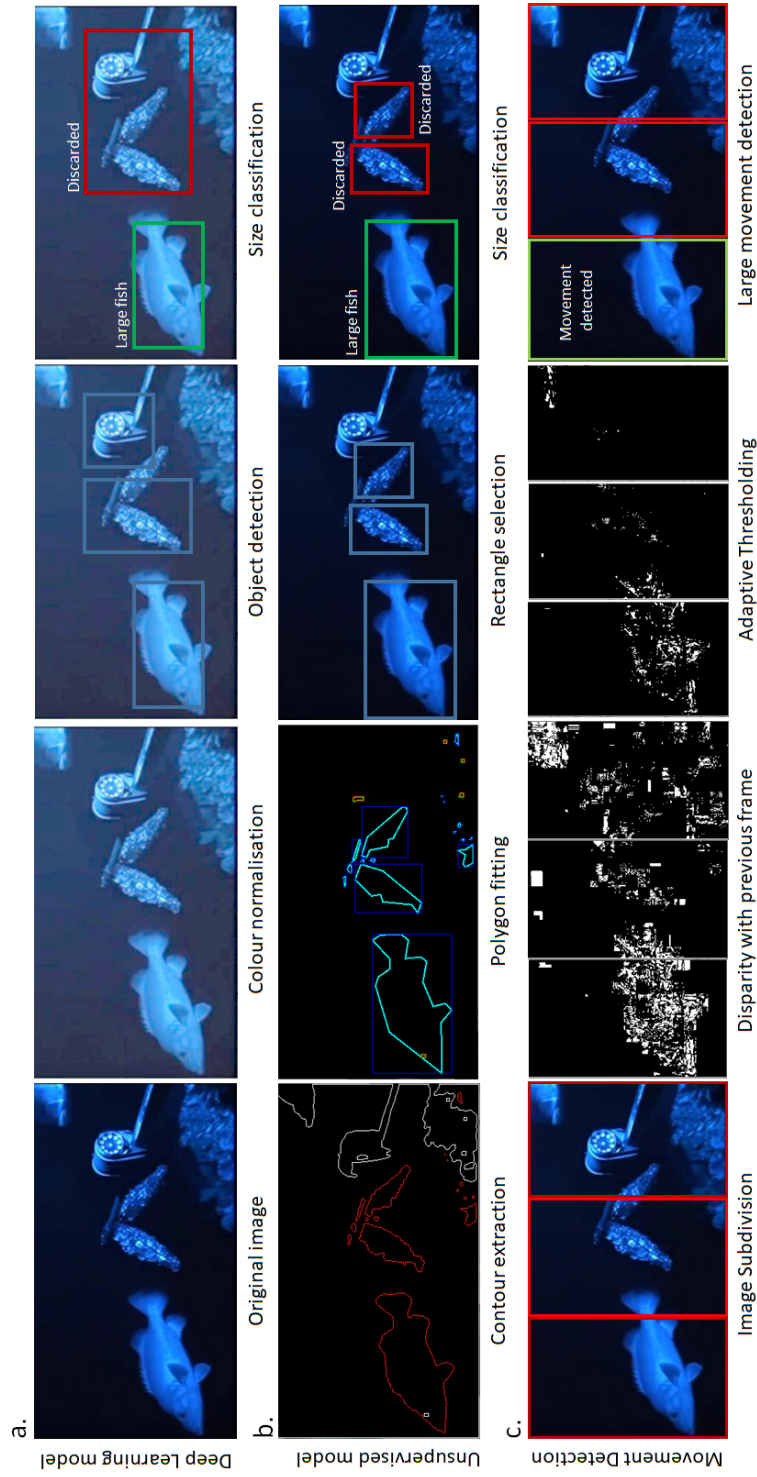


Figure 2: Comparison between the three workflow variants of our methodology to detect large fishes: a) Deep Learning-based object detection pipeline with distance and size classification; b) Unsupervised model with distance and size classification; and c) movement detection model with temporal adaptive threshold.

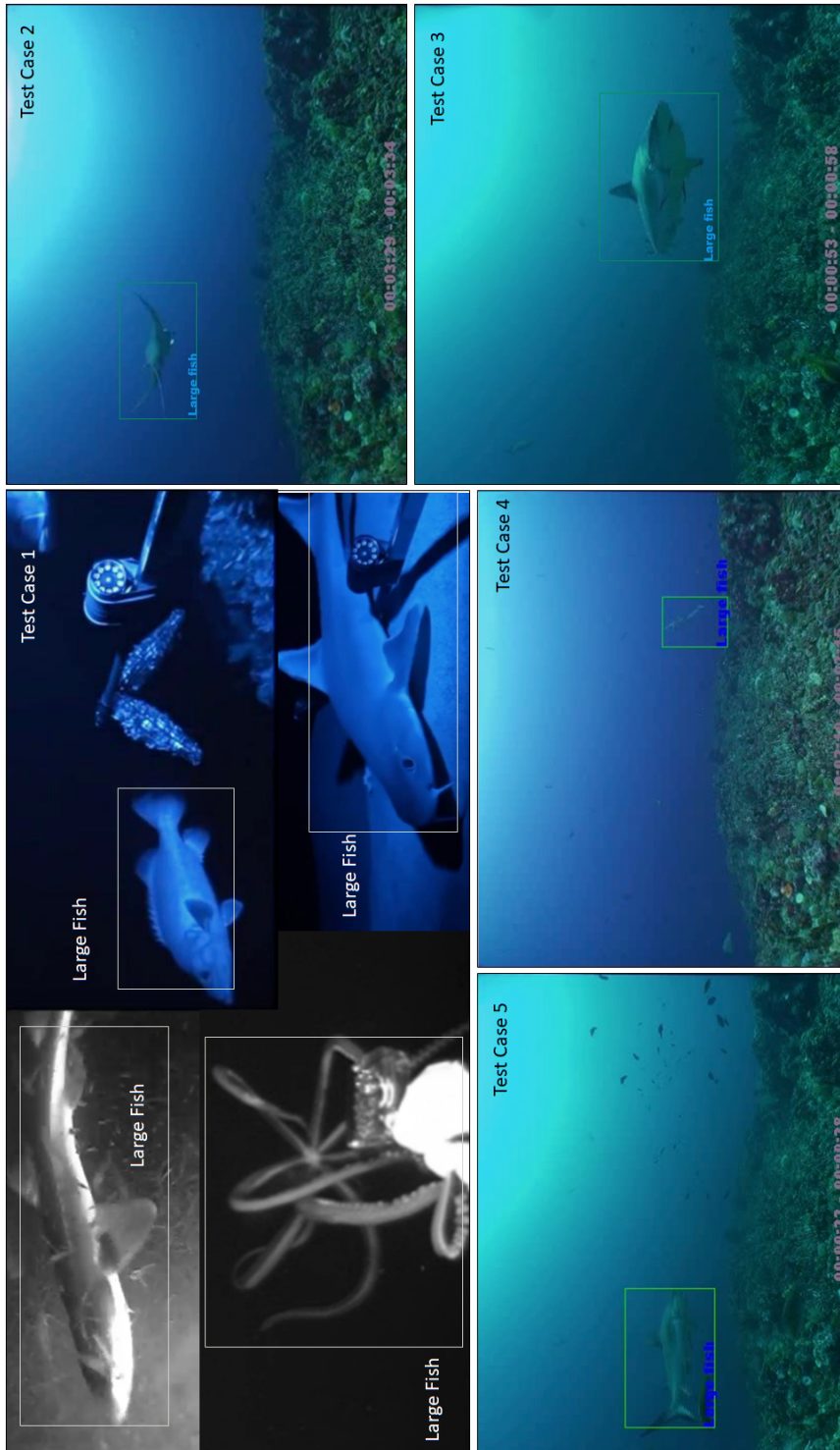


Figure 4: Examples of events in our test case videos of large fishes passing in front of the camera and correctly captured by our workflow.