Fish4Knowledge Deliverable 5.5

Experimental evaluation report 2

Principal Authors:	Bastiaan J. Boom, Concetto Spampinato, Simone Palazzo, Emmanuelle Beauxis-Aussalet, Xuan Huang, Gayathri Nadarajan, Cheng-Lin Yang
Contributors:	UEDIN
Dissemination:	PU

Abstract: The second experimental evaluation report describes the progress made on the different software components and the entire system with respect to the first experimental report (Deliverable 5.4). In this report, the separate components are evaluated (where possible) with respect to the performance achieved in the first experimental report. Currently, our system has processed all the videos in the database with the fish detection and tracking software and 271722 (51.4%) videos are processed with the fish recognition software. In the current videos, we discovered that many videos suffered from video encoding errors and blur due to dirty cameras. Our system is able to prioritise for videos without these errors. Improvements to the user interface allow users to visualise more data and make it easier to visualise this data. The workflow allows users to execute VIP (Video/Image Processing) software on the super computers, giving them the ability them to analyse some of the videos with other software versions and verify their observations.

Deliverable due: 36 Month

1 Introduction

In the Fish4Knowledge project, we had developed a fully working prototype system at the end of the second year of the project. In the final year of the project, this system is extended with new version of the different components that make up the system and we connected all the major components with each other, obtaining a fully working prototype system.
In this report, we will discuss the improvements of the system compared to the previous system evaluated in Deliverable 5.4. In this document, the evaluation is performed for each component in the system separately. The creators of the components are also responsible for the evaluation, because they are the experts in the different computer science areas which are brought together to obtain the overall system. The different components will be briefly discussed together with the evaluation that has been performed on each of them.
The overall system is also evaluated, however the most visible part of this system is the user interface. Although the user interface should also give an idea of the overall system, it is difficult for users to evaluate the underlying workflow methodology of the system, because this is hidden from the users. This means that a full evaluation of the system by the user (marine biologists/ecologists) will never be possible. However the user interface is an ideal tool to obtain feedback on the entire system, and then improve the underlying processes. After discussing the user interface, we will also give an idea of the amount of data that has been processed by the system. This data is currently organised in such a way that users can query it with the user interface. We do not know of any project that processed such amounts of video data, making Fish4Knowledge a truly unique project.
This deliverable is organized as follows. In Section 2 the state of the system will be discussed. Afterwards, the evaluation of the different components are separately described: Fish Detection, Tracking and Recognition in Section 3, computation time and stability of VIP in Section 4, Workflow in Section 5 and the User Interface in Section 6. In Section 7, conclusions will be given, as well as suggestions for further improvements that could be done as a continuation of this (or a similar) project.

2 State of System

The Fish4Knowledge project has developed a fully working prototype system (see `http://gleoncentral.nchc.org.tw/`). This was already reported in Deliverable 5.4, however some components were not yet fully connected with each other and newer versions of the components are created based on feedback from marine biologist, and on the domain knowledge and new insights we acquired. This section gives an overview of the improvements of the system described in Deliverable 5.4, from the perspective of the entire system. The improvements on the component level will be mentioned quickly and will be discussed in more detail in the next sections of this document.
In Deliverable 5.4, we reported that some components were not fully connected with each other (i.e., workflow and interface). This connection has now been achieved, allowing marine biologists to run video processing software on the videos in the database. Secondly, we reported that the collected groundtruth evaluation data was not available to the user interface, because the websites that allowed us to create the groundtruth data were hosted in Europe. Hosting the websites in Europe was necessary to provide speedy interaction with the main users located

Europe. Currently, most of the collected groundtruth evaluation data is stored in the database in Taiwan. The format of the data defined/agreed by the User Interface developers allows them to inform/educate users about the uncertainty coming from automatic video processing software. Large improvements are also achieved on the individual components. The video processing is able to classify videos into categories like “blurred”, “normal”, “encoding problem”, etc. We discovered surprising video analysis results which, by checking the original videos, were due to for instance “encoding problem”, where often more fish are detected then one would normal expect. For the fish recognition components, we are able to recognise more species, going from 15 to 23 species. Also, the recognition component can filter out false positives from the detection stage. The user interface is improved in both the usability and the fact that it can present more views on the data. There is also a connection to the workflow giving marine biologists the ability to process videos. The workflow is able to achieve more stability by rerunning jobs and can give estimates of the execution times. The detail about the improvements of the different components can be found in the following sections.

3 Video Analysis Components

3.1 Video Classification

One important component of the underwater video analysis system is video classification, as it provides information on the overall quality of the processed videos, which is necessary to interpret the achieved results. The video classification component uses the Bag of Features (BoF) approach and works at video level, i.e., given an input video, a model vector sequence is obtained and its classication as a specific video class is carried out by means of SVMs operating on SIFT descriptors of image corners extracted from a subset (20 in our case) of video frames (more details are given in Deliverable 1.2).

We tested our approach on a set of about 130 videos (20 frames uniformly distributed in each video), manually labeled into 6 classes (see Fig. 1 for some examples): “Algae”, “Blurred”, “Complex Scenes” (in terms of frame textures), “Encoding Errors”, “HighlyBlurred”, “Normal”.

The accuracy was measured in terms of precision/recall curves, which are shown in Fig. 2. It is possible to notice that the best results were achieved for “HighlyBlurred” and “Complex Scenes” classes because their features lie in regions of the feature space pretty much separated from the ones of the other video classes. The lowest performance (still high, about 0.88 of F-measure) was obtained with the “Algae” and “Encoding” classes for two different reasons: 1) the “Algae” was often misclassified as “Blurred” (not vice versa) because in both cases the clearness of the image is low, 2) the videos from the “Encoding” class do not have all the frames corrupted and since our approach uses only a subset of frames it may happen that extracted frames are uncorrupted. As of July 2013, we have classified 534,589 videos, so divided: Algae: 47,003 (8.8%), Blurred: 175,182 (32.7%), Complex Scenes: 36,948 (6.9%), Encoding: 131,533 (24.7%), HighlyBlurred: 63,505 (11.9%), Normal: 74,266 (13.9%). 6,152 (1.1%) videos were not classified. By looking at the classification results, it is possible to notice that about 85% of the whole videos were of very low quality and this demonstrates the actual complexity to deal with “real-life” underwater videos.

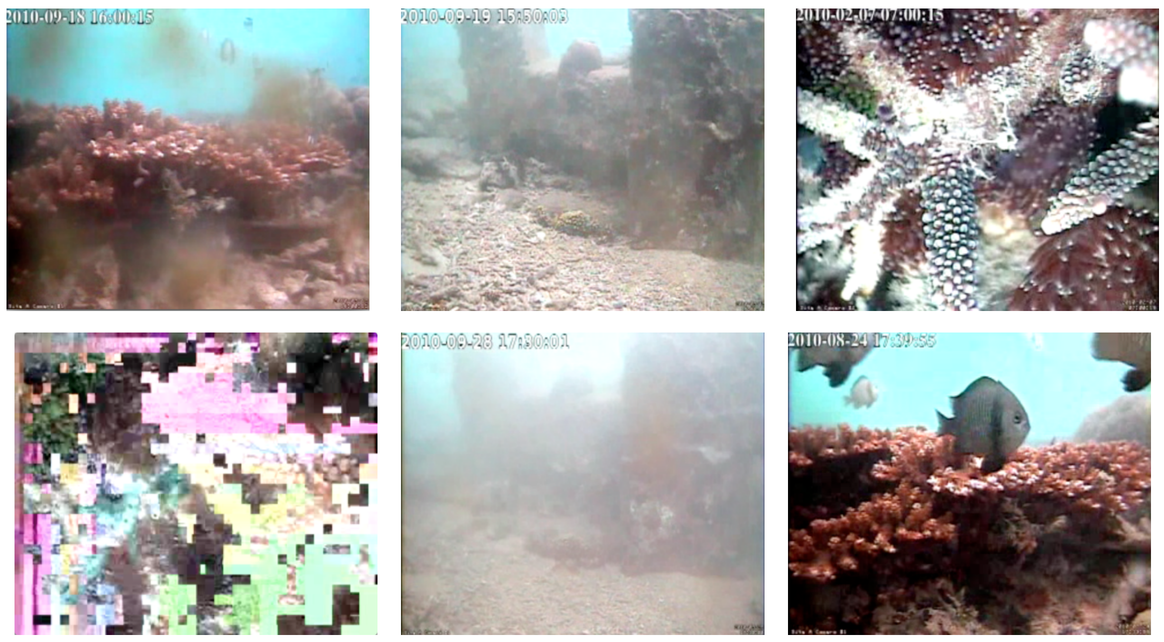


Figure 1: Video Classes (from top-left to bottom-right): 1) Algae, 2) Blurred, 3) Complex Scenes, 4) Encoding, 5) HighlyBlurred, 6) Normal

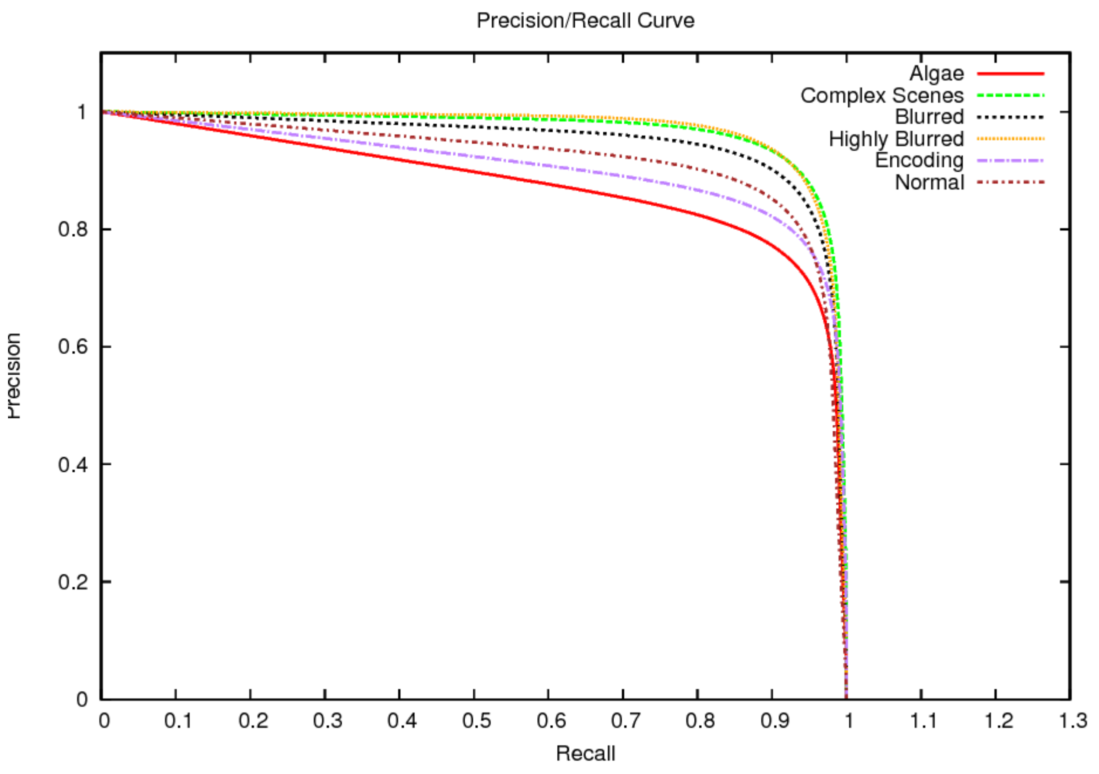


Figure 2: Precision/Recall curves per video class

3.2 Fish Detection

Here we report the results of a new detection component and compares its performance to the one achieved by several state-of-the-art approaches on two datasets: the F4K underwater dataset

and the I2R dataset [8]. The reason of testing the new component with different targets was to demonstrate its ability to generalize over multiple domains[1].
The new detection component relies on a joint domain-range approach [18] which models background and foreground by employing object textures in the modeling process. This component, therefore, differs significantly from the approaches described in previous deliverables in that it models not only the background pixels but also the foreground ones and exploits complex texture features, namely, the textons as they have proved to be robust against illumination changes. We did not use any post-processing to improve detection results but removed the connected components having area less than 15 pixels. The accuracy was measured in terms of precision/recall curves and F-measure F_1, defined as:

$$Precision = \frac{TP}{TP+FP}, \;\; Recall = \frac{TP}{TP+FN}, \;\; F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (1)$$

where TP, FP and FN are, respectively, the true positives, the false positives and the false negatives measured when comparing, on a pixel basis, the ground truth binary masks and the output masks of the background modeling approach.
For testing the performance of the new approach on the underwater domain, we used a dataset consisting of 17 “real-life”underwater videos. This dataset, in detail, contains about 3500 fish masks, manually labeled using the tool in [7] and it is available at `http://f4k.dieei.unict.it/datasets/bkg_modeling/`, equally balanced (as per reviewers’ suggestion) between seven video classes (classification[2] performed according to the typical features of underwater videos): “Blurred” (smoothed and low contrasted images), “Complex Background Texture” (background featuring complex textures), “Crowded” (lof of fish), “Dynamic Background” (background movements, e.g. plants movements etc.), “Hybrid” (more than one features: e.g. plant movements together with luminosity changes), “Luminosity Change” (videos affected by transient and abrupt luminosity changes), “Camouflage Foreground Object” (e.g. fish with colours similar to the background) and examples are shown in Fig. 3. We did not use the same dataset of Deliverable 1.1 because it was unbalanced among the video classes (see Deliverable 1.1) and this dataset is an extension of the one presented in Deliverbale 5.4 in order to include more challenging cases where to test our background modeling approaches.

The first evaluation aimed at understanding how our new approach performed on the different scenarios that may happen in underwater monitoring. Fig. 4 shows the Precision/Recall curves achieved for each video class. It is possible to notice that our method performs the best on blurred videos (because of the simplicity of the background), whereas its performance dropped with background object movements (“Dynamic Background” class), though the performance in that case showed an average F-measure of about 0.74.

Then, we compared the results of our method to the same approaches used in D5.4, namely:

- P-Finder [22] which models the background with only one single Gaussian *pdf* (Gaussian):

[1]The approach was tested also on the Background Models Challenge (BMC) dataset [20] but the results of the challenge are not available yet.

[2]Please note that this video classification is different from the one adopted for video classification (see previous section) as it is meant to test background modeling approaches on different conditions that cannot be automatically classified using visual descriptors at the frame level

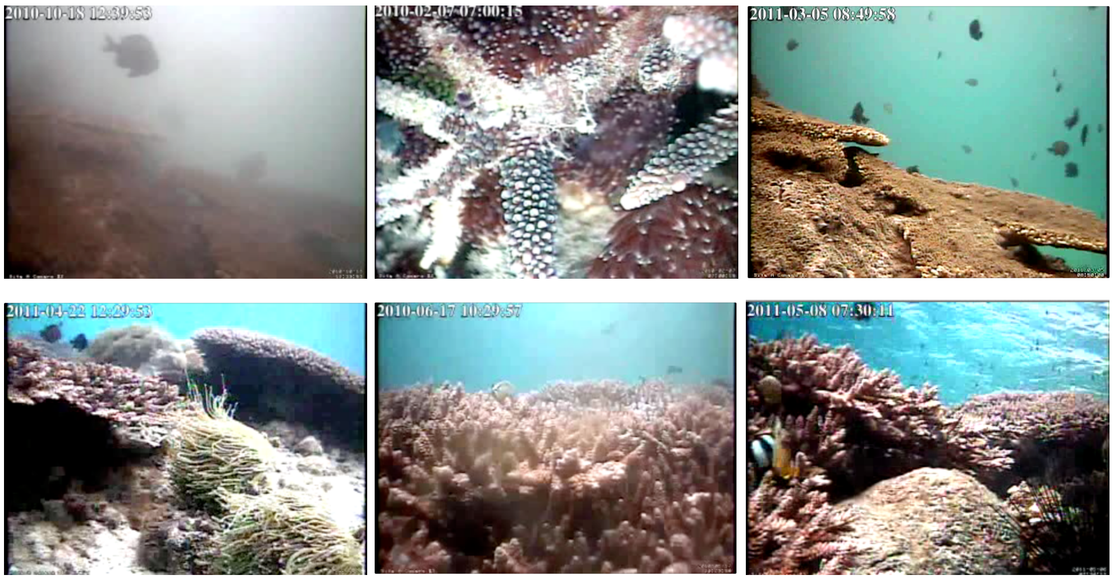


Figure 3: Underwater Video Dataset. From top-left to bottom-right: 1) Blurred, 2) Complex Background Texture, 3) Crowded, 4) Dynamic Background, 5) Luminosity Change, 6) Camouflage Foreground Object

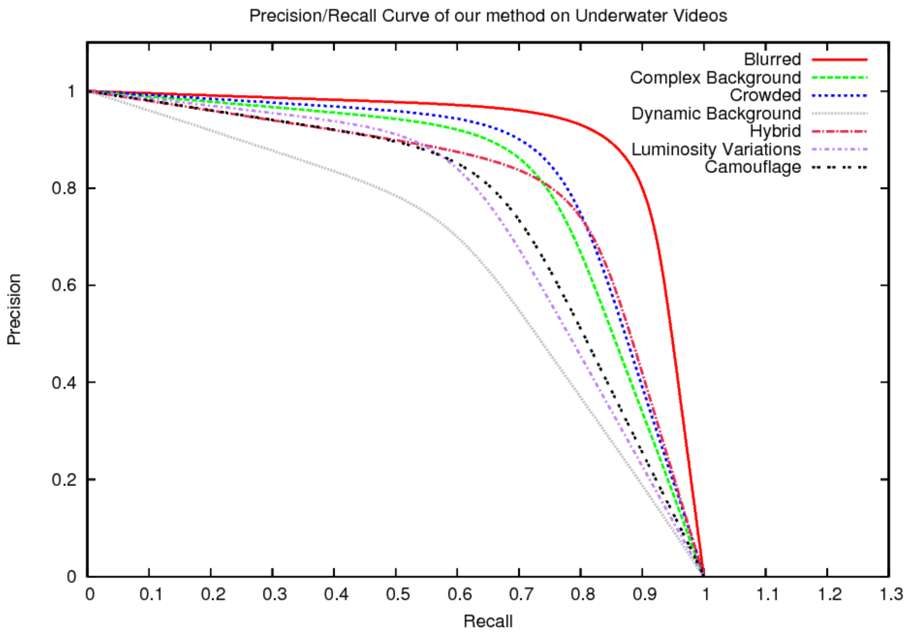


Figure 4: Precision/Recall Curves of our method with underwater videos

Video Class	P-finder	GMM	$ZGMM$	$EIGEN$	$ML - BKG$	$KDE - RGB$	$VIBE$	*Our Method*
Blurred	78.40	79.23	81.32	80.42	73.41	90.06	86.30	92.15
Complex Background Texture	69.73	71.48	68.17	75.27	76.85	66.64	74.17	80.37
Crowded	72.85	75.32	75.56	73.65	79.83	81.72	86.83	79.84
Dynamic Background	39.92	48.23	54.48	58.99	80.60	56.78	57.98	73.41
Hybrid	64.86	65.86	66.89	76.34	77.38	78.88	73.56	84.56
Luminosity Changes	54.15	65.84	64.45	63.19	61.07	71.47	72.92	74.43
Camouflage Foreground Object	67.90	72.42	67.68	66.20	77.43	57.72	72.88	80.36
Average	63.97	68.34	68.36	70.58	75.22	71.89	74.94	80.73

Table 1: Average F-Measure scores (in percentage) for different methods on our underwater dataset.

- Two methods that exploit mixture of Gaussians, namely, the original Gaussian Mixture Model by Stauffer and Grimson [19] (*GMM*) and its improvement by Zivkovic (*ZGMM*) [24];
- The Eigenbackground (*EIGEN*) Subtraction method [12],
- Two non-parametric kernel density estimation approaches: Sheikh's method [16] (*KDE-RGB*), which uses only colour features for modeling the background, and the MultiLayer background model (*ML-BKG*) by Yao in [23], which, instead, employs also texture features computed via Local Binary Patterns.
- *VIBE*-like approach based [2], which was used for production run because of it balances efficiency and accuracy.

The performance of the different methods are reported as F-Measure values and illustrated in Table 1, which shows that combining colour and texture features (as in our method and in *ML-BKG*) enhanced the background modeling's performance also in complex scenarios where targets and background have similar texture features. Unlike *ML-BKG*, which performed well with complex background due to the high texture variability, our approach shows good accuracy also in the cases of smooth regions (e.g. Blurred class) because of textons' ability to describe also tiny texture variations.
A qualitative comparison of our approach, *VIBE* and *ML-BKG* is presented in Fig. 5, which shows that our approach had good qualitative performance when compared to the other two. Also in this case, the increment in accuracy was achieved at the expense of efficiency; in fact, the average number of frames (size 320×240) processed per second for *VIBE*, *ML-BKG* and our approach were, respectively, 100 frames/sec, 20 frames/sec and 1.5 frames/sec with a C++ implementation running on a PC powered by an Intel i7 3.4 Ghz CPU and 16GB RAM.

Then another evaluation was carried on the I2R Dataset to analyse how the approach performed on scenarios different from the underwater one. The I2R Dataset contains nine videos (frames 120×160) showing people in real-life scenarios; the ground truth was hand labeled, consists of 20 labeled image per video and is available at `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`. Examples of these videos are shown in Fig. 6. Table 2 compares the F-measures achieved by our method to the ones obtained by recent and powerful KDE state-of-the-art approaches, namely, the one relying on a joint domain-range approach which uses only colours [16] (referred as KDE-RGB), the one using texture features computed through scale-invariant local ternary patterns (SILTP) [9] and the ones combining textures with colours: VKS rgb and VKS Lab plus SILTP [11]. Although in some cases (e.g. Trees) our approach performed slightly worse than the other methods, on

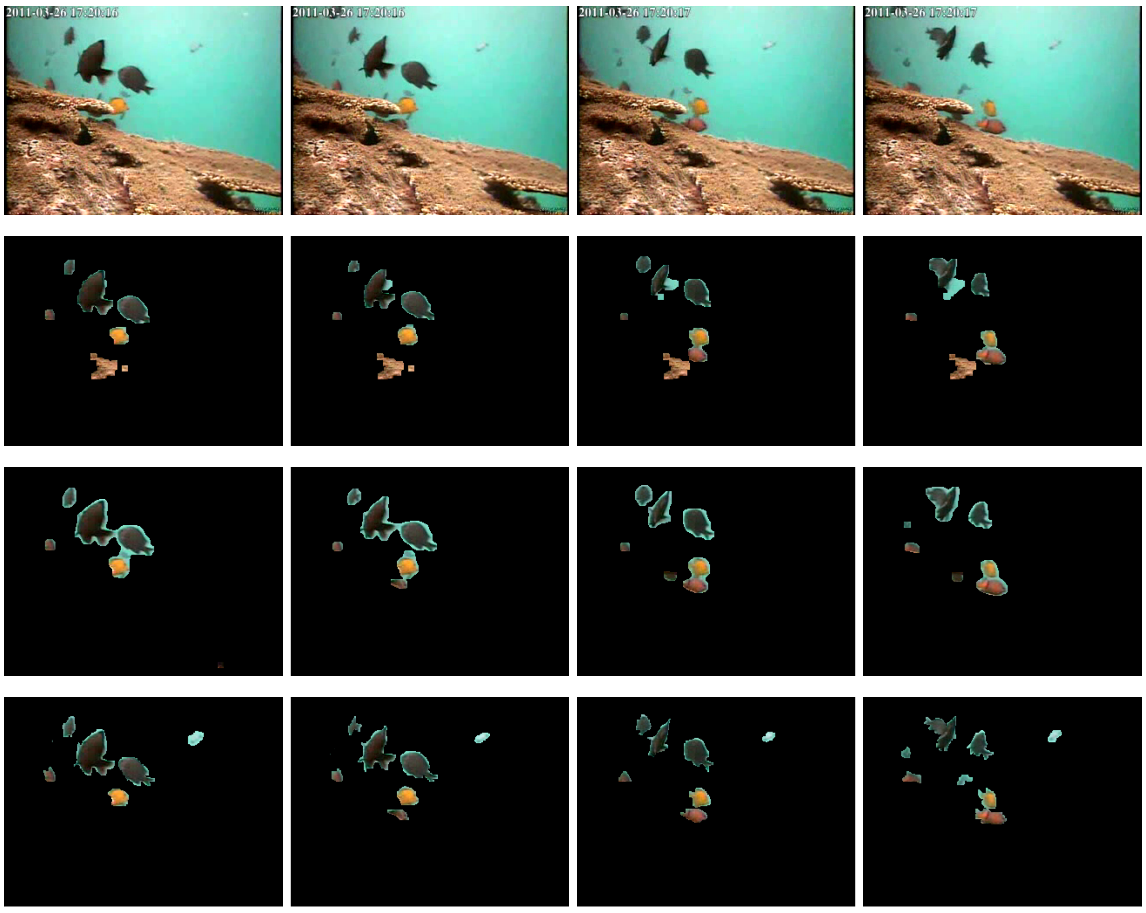


Figure 5: Qualitative Comparison: background subtraction results with (from top to bottom) 1) $VIBE$ (second row) which detects parts of rocks as fish because of light changes, 2) $ML-BKG$ (third row) which outperforms $VIBE$ in term of false positive reduction, in fact, it did not detect the rocks, which, instead, were misclassified as foreground by $VIBE$ and 3) our approach (last row) which is able not only to reduce false positives (rocks not detected) but also to detect tiny fish (bright spot on the right hand side) whose appearance looks like the background's one.

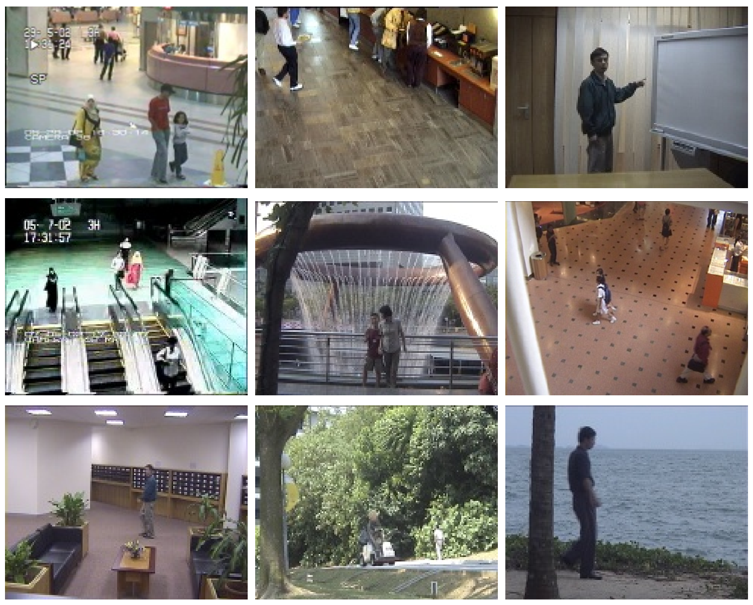


Figure 6: I2R Dataset. From top-left to bottom-right: 1) AirportHall, 2) Bootstrap, 3) Curtain, 4) Escalator, 5) Fountain, 6) ShoppingMall, 7) Lobby and 8) Trees, 9) WaterSurface

Video	KDE-RGB [16]	SILTP [10]	VKS rgb [11]	VKS Lab plus SILTP [11]	Our Method
AirportHall	61.34	68.02	70.44	71.28	69.23
Bootstrap	74.64	72.90	71.25	76.89	76.47
Curtain	97.73	92.40	94.11	94.07	94.89
Escalator	65.41	68.66	48.61	49.43	72.02
Fountain	51.32	85.04	75.84	85.97	83.21
ShoppingMall	60.36	79.65	76.48	83.03	78.54
Lobby	67.79	79.21	18.00	60.82	66.34
Trees	66.75	67.83	82.09	87.85	81.89
WaterSurface	81.57	83.15	94.83	92.61	92.51
Average	69.66	77.43	70.18	77.99	79.46

Table 2: Average F-Measure (in percentage) for different methods with the I2R datasets.

average it outperformed them. The motivation behind this good performance may probably be found in the capabilities and robustness of the textons to describe image textures.

The last evaluation aimed at analysing how background modeling algorithms performed on different regions of the recorded scenes and how a suitable combination (made according to video class and scene region) of them may achieve improved performance. For this reason, we also labeled regions in the underwater video dataset described earlier; in particular, for each ground truth video we also labeled, in the first frame without objects, four areas (which represent four different levels of - increasing - complexity for background modeling): “Open Sea”, “Corals” and“Rocks”[3]. We compared (see Table 3 for the results) all the previous approaches on the different image zones with the exception of the new component and the $KDE-RGB$, because of their limited efficiency. Finally, Table 4 shows which is the best algorithm per image region and video class. These results provide two important information: 1) a suitable

[3]Please note that the video frame labeling was not exhaustive, i.e. only some regions were labeled because of the variability of the analysed scenes. This implies that the results in Table 4 may not be derived from the ones in Table 1 as the latter are obtained when analysing the whole video frames, while the former only some regions

Image Region	P-finder	GMM	$ZGMM$	$EIGEN$	$ML-BKG$	$VIBE$
Open Sea	78.18	79.40	80.59	79.84	83.03	85.95
Corals	61.03	59.28	54.34	66.02	75.03	61.27
Rocks	64.44	73.90	68.83	64.97	76.60	77.17

Table 3: Average F-Measure scores (in percentage) for different methods per image region.

combination of background modeling approaches allows for improved performance (5% on average comparable to our new approach); 2) they justify (beyond the processing times shown in Table 5) the adoption of $VIBE$ for production run; in fact, it generally performs the best both on blurred videos (which account to about 50% of the totality of the F4K videos, see the "Blurred", "HighlyBlurred" values in the section on video classification) and on the open sea area where most of fish appear (about 70% of the labeled fish is in that region). Table 5 reports the processing times (frames/sec) of the background modeling algorithms (C++ implementation) on a PC powered by an Intel i7 3.4 Ghz CPU and 16GB RAM.

Video Class/Image Region	Open Sea	Rocks	Corals	Average
Blurred	VIBE(91.45)	–	VIBE (66.94)	**79.19**
Complex Background Texture	VIBE(79.95)	ML (86.64)	ML (87.37)	**84.65**
Crowded	VIBE (88.67)	ML (80.32)	–	**84.49**
Dynamic Background	VIBE (82.10)	ML (83.11)	ML (86.98)	**84.06**
Hybrid	EIGEN (80.16)	–	ML (77.14)	**78.65**
Luminosity Changes	VIBE (85.95)	ML (77.14)	ML (84.68)	**82.59**
Camouflage Foreground Object	VIBE (88.52)	ZGMM (86.29)	GMM (65.22)	**80.01**
Average	**85.25**	**82.70**	**78.05**	

Table 4: Best performance (in terms of F-Measure) per video class and image region. The missing values mean that for that specific video class (i.e. for the ground truth videos), the region was not available.

3.3 Fish Tracking

In the first evaluation report (Deliverable 5.4) we presented the results we obtained for the three candidate tracking algorithms – the covariance-based tracker [14], CAMSHIFT [3] and CONDENSATION [6] – and motivated our decision for choosing the covariance-based tracker for production run on all historical video clips.

In the following, we will rediscuss the main features of this approach and introduce an extended version of the algorithm, designed to tackle some of the problems found in the original version and to include the suggestions provided by the reviewers.

3.4 Covariance tracking

The idea of covariance tracking is inspired by the work in [14] and further developed in [17] and consists in encoding the spatial, statistical and appearance properties of an object in a video by means of the covariance matrix of a set of pixel-based feature vectors (including location coordinates, colour information, intensity derivatives). This model represents a compact, efficient and elegant approach to describe the appearance of objects, such as fish, which continuously undergo changes in shape/size and occlusions.

However, the testing and review phases of the algorithm showed several important limitations:

Algorithm	320 × 240	640× 480
P-Finder	250	60
GMM	200	50
VIBE	100	25
ZGMM	100	25
EIGEN	30	10
ML-BKG	20	3
Our recent approach	1.5	–

Table 5: Processing Times (frames/sec) of the employed background modeling algorithms.

- Since the algorithm only performed associations between the blobs produced by the fish detection module, its accuracy was necessarily dependent on the latter's. A failure in the fish detection algorithm necessarily propagated to the fish tracking algorithm;
- The computation of the search area for a fish in a new frame was performed through a heuristic method, based on the history of objects in the video, rather than on a more generic model;
- The algorithm could not handle occlusions, since two fish "touching" (from a bi-dimensional perspective) each other would be identified as a single blob by the fish detection algorithm, which on one hand caused the tracking algorithm to associate the aggregated blob to one of the two fish (thus missing the other one), and on the other hand caused the associated fish's model to be disrupted by the inclusion of pixel information belonging to the other fish. However, in videos where the image quality was acceptable (i.e. enough to distinguish the patterns of the occluding fish), the covariance method was able to recover fish after occlusions; this means that the occlusions produced sometimes an overcounting which was then corrected a-posteriori).

3.5 Covariance particle filter

The *particle filter* framework [6] models location and motion information of an object by a set of state vectors (called *particles*) weighted by the similarity between the object's model and the visual features of the area described by the corresponding particle. This approach allows to specify a mathematical motion model (whose complexity depends on the targets' motion pattern) which defines the way particles move (as well as, implicitly, the object's area), while at the same time allow for significant random variation of the particles, in order to explore the nearby regions.

This algorithm presents a strong break point with respect to the previous one in that it uses information from the upstream detection module only as a hint for where fish might have moved to, rather than as the only source of possible locations. The advantages of this modifications can be found in the way the tracker behaves when the fish detection algorithm fails:

- If a camouflage effect hides a fish due to its similarity to the background, the tracker still tries to locate it, in spite of the absence of detections.
- If two fish occlude, the particle filter tear them apart, whereas the previous tracker only saw them as a single blob.

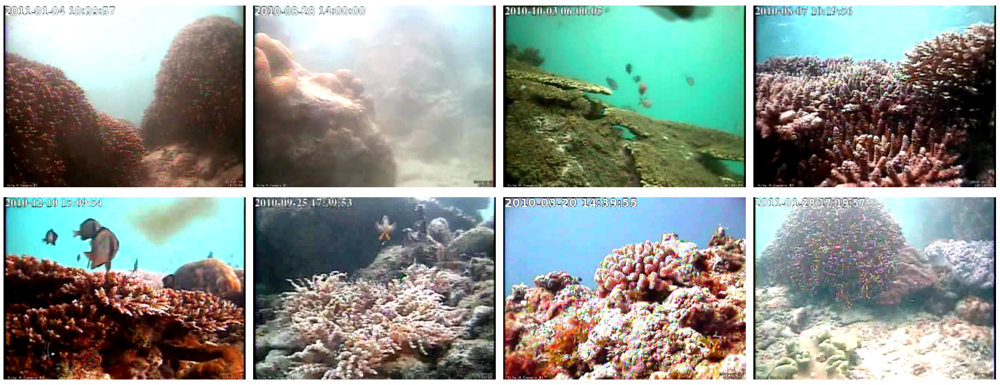


Figure 7: The scenes included in the tracking ground truth video set.

3.6 Performance evaluation

The comparison between the original covariance-based tracker and the covariance-based particle filter was performed on a set of 11 ground truth videos. Each video is 10 minute long, sampled at 8 frames per second; the resolution is 320×240 (for 7 videos) and 640×480 (for 3 videos). The ground truth annotation process consisted in manually drawing the contours of all fish in each frame and associating detections in different frames as instances of the same fish. In total, the annotated dataset contains 35391 single detections for 2218 trajectories. Figure 7 shows the kind of scenes included in the ground truth.

Before looking at the results, we should note that, unlike the evaluation presented in Deliverable 5.4, where the input detections to the trackers were the manually annotated objects (in order not to let detection mistakes affect the tracker accuracy), in the present evaluation we used the outputs of the fish detection (in these experiments, VIBE), to analyse how the tracker performed with noisy objects.

As in Deliverable 5.4, the tracking accuracy is shown in terms of the following indicators:

- Correct Counting Ratio (*CCR*): percentage of correctly identified ground-truth fish. This ratio provides information not only on the tracking algorithms but also on the overall system performance from background modeling to fish detection to tracking.
- Average Trajectory Matching (*ATM*): average percentage of common points between each ground-truth trajectory and its best matching tracker trajectory;
- Correct Decision Rate (*CDR*): let a "tracking decision" be an association between a fish at frame t_1 and a fish at frame t_2, where $t_1 < t_2$; this tracking decision is correct if it corresponds to the actual association, as provided by the ground truth. The correct decision rate is the percentage of correct tracking decisions, and gives an indication of how well the algorithm performs in following an object, which is not necessarily implied by the average trajectory matching (see Deliverable 1.1 for details).

Table 6 shows the results of the original covariance-based algorithm (labeled *COV*) and the new covariance-based particle filter (labeled *COVPF*) on each of the 11 videos in the dataset, and on average for all videos.

It is possible to notice a decrease in the ATM score, which can be explained by the presence of more false positives in the trajectory (due to the particle filter's attempts at finding detections

		COV			*COVPF*		
Video	**Objects**	**ATM**	**CCR**	**CDR**	**ATM**	**CCR**	**CDR**
1	1058	0.75	0.70	0.74	0.50	0.68	0.93
2	3072	0.92	0.51	0.81	0.84	0.53	0.93
3	16321	0.66	0.67	0.77	0.56	0.65	0.65
4	1927	0.73	0.56	0.80	0.69	0.55	0.89
5	1284	0.64	0.59	0.67	0.48	0.59	0.78
6	1656	0.70	0.55	0.66	0.56	0.52	0.87
7	5477	0.66	0.72	0.75	0.71	0.74	0.77
8	820	0.95	0.90	0.73	0.80	0.80	0.75
9	1447	0.88	0.66	0.73	0.84	0.63	0.83
10	1903	0.84	0.57	0.70	0.80	0.53	0.75
Avg		*0.77*	*0.64*	*0.74*	*0.68*	*0.62*	*0.82*

Table 6: Evaluation of the original covariance tracker and the new one using the particle filter approach. For each video we show the number of objects included in the corresponding ground truth and the three tracking evaluation indicators for the two algorithms; the last row shows the average values of all scores.

		COV			*COVPF*		
Video	**Objects**	**ATM**	**CCR**	**CDR**	**ATM**	**CCR**	**CDR**
1	344	0.86	0.85	0.84	0.86	0.85	0.88
2	260	0.84	0.85	0.83	0.85	0.85	0.88
3	121	0.75	0.71	0.81	0.80	0.76	0.83
Avg		*0.81*	*0.80*	*0.83*	*0.83*	*0.82*	*0.86*

Table 7: Evaluation of the algorithms on the HD-quality AQUACAM videos.

for a fish in the absence of motion data), a general constancy in the CCR score, and an increase in the CDR score. This last result reflects the improvements introduced by the particle filter: when a fish swims “alone”, i.e. without fish nearby (which may disturb the tracking association process), the only source of tracking decision errors is missing fish detections; instead, in the case of occlusions, the probabilities of misassociations increase, as the tracker has multiple candidates to associate to each tracked fish. In order to evaluate the effect of the videos’ low frame rate on the quality of the tracker, we also tested the algorithms on a set of preliminary videos from the AQUACAM project[4] (in collaboration with the F4K Research Consortium). Such videos have a much higher resolution (1920×1080 pixels), bitrate (15 Mbps) and frame rate (29 frames per second) than the ones available for this project; an example frame is shown in Figure 8. Our results, run on 3 such videos, are shown in Table 7, and show the benefits that both trackers obtain from the higher frame rate and video quality. In particular, the particle filter tracker showed the largest improvements. This is due to the fact that smaller motion between frames yields a more stable motion model, which allows to distribute the particles more effectively in the search region. This results in higher quality tracking, as is shown by the corresponding values for trajectory matching and correct decisions.

[4]http://c-fish.org/what-we-do/aquacam-research-programme/

Figure 8: Example of scene from the AQUACAM video dataset.

3.7 Fish Recognition

3.7.1 Fish recognition with reject option

We extended the Balanced-Guaranteed Optimized Tree (BGOT) with a reject option to filter less confident recognition results (shown in Figure 9). There are three types of samples that the reject function is designed for. The first type, non-fish objects (coral, seaweed, *etc*), is due to false positives of the fish detection/tracking software. Ideally, the acquired inputs for recognition only contain fish image and its detected boundary. However, due to the complexity of underwater environment (*e.g.* light distortion, fish occlusions and illumination transformation), the input data contains false positive detections and incomplete fish images. These inputs cause unexpected results with the recognition component. The second type is from the misclassified samples of BGOT hierarchical tree. As the hierarchical classification accumulates errors along its decision path, the rejection system provides an alternative channel to discover the uncertain decisions at the leaves of the classification hierarchy. The third type is for new species of fish. Our recognition algorithm covers top 23 species which is over 95% proportion of the observed fish. However, more than 2000 species of fish live in the Taiwan sea. These species are not included in our ground-truth dataset. We use the reject option to probe these new species of fish which would benefit the marine biologists. In practice, we apply a Gaussian Mixture Model (GMM) as the reject option to evaluate the posterior probability of the fish images, and reject those that are non fish objects, false species recognition, or new species of fish.

3.7.2 Evaluation

A GMM model for each species is trained with a subset of features using the forward sequential selection method. For each probability density function $P(X \mid C)$, we use the features X obtained for class C_i by the feature selection and estimate the underlying GMM according to those features. At each tree node, the rejection classifier is given a by the $P(C \mid X)$, the probability that the sample is actually class C according to the GMM likelihood score. The diversity improves the performance of the hierarchical classification, *e.g.*, learn a threshold from

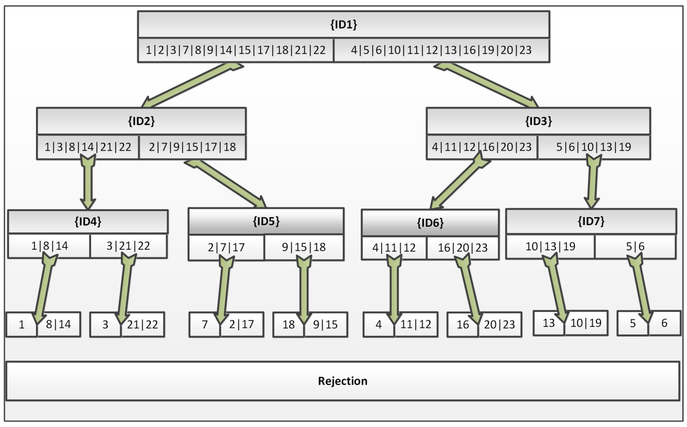


Figure 9: Result rejection in fish recognition.

the training samples and reject test samples whose likelihood scores are below the threshold. The rejection uses the *a posteriori* probability given the observed features X with the predicted class C_i:

$$p(C_i \mid X) = \frac{p(C_i)p(X \mid C_i)}{p(X)} = \frac{p(C_i)p(X \mid C_i)}{\sum_i p(C_i)p(X \mid C_i)} \quad (2)$$

The prior knowledge $p(C_i)$ is calculated from the dataset.

We evaluated the reject option in our fish recognition software. For each fish species, we trained a GMM with the selected feature subset obtained using forward sequential selection method. We used unsupervised learning [4] to select a mixture model where the maximum number of Gaussian density component is sets to 7. The experiment is carried out by comparing our GMM-based method with two state-of-the-art methods: 1) relating SVM outputs to probabilities [13], and 2) soft-decision hierarchical classification with a reject option [21]. Since our database is imbalanced and only the top species have adequate samples to train the rejection model, we only apply the reject option to the top 6 species. The data is acquired from the fish detection/tracking results with 24150 fish images of the top 15 most common species. The minority (untrained) species (8 species, 3220 fish images) are also included in the test set to test the performance in rejecting new species.

Algorithm	AP (%)	AR (%)
BGOT baseline [5]	56.51	**91.07**
BGOT+SVM probabilities [13]	58.96	90.93
BGOT+soft-decision hierarchy [21]	58.91	90.70
BGOTR	**65.02***	88.31

Table 8: Fish recognition result. * means significant improvement with 95% confidence by t-test.

The experiment result Table 8 demonstrates that our method rejects a portion of the misclassified samples (significant improvement in AP) while the cost is that it also rejects a small proportion of correctly classified samples (small reduction in AR). We compare it to two other rejection algorithms [13, 21] and our method achieves significant better performance in AP. The proposed method improves BGOT hierarchical classification in two aspects: 1) filters out part of the misclassified samples, and increases the averaged precision with a small reduction of the average recall; 2) finds potential new samples which do not belong to any known classes. It detects a set of samples which have higher probability of coming from new species, and therefore, reduces the work for finding new fish, especially in a large database of underwater videos. To summarise our result, we use F-score to consider both the average recall and the average precision of the test. The general formula of F-score for a positive real β is:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3}$$

We use the F_1 measure, which is the harmonic mean of precision and recall, as shown in Table 9.

Algorithm	F_1-score
BGOT baseline [5]	0.7135 ± 0.0227
BGOT+SVM probabilities [13]	0.7150 ± 0.0222
BGOT+soft-decision hierarchy [21]	0.7140 ± 0.0225
BGOTR	**0.7485 ± 0.0194** *

Table 9: F-score result. * means significant improvement with 95% confidence.

3.7.3 Conclusion

We provided a reject option by applying the GMM at the leaf nodes of the BGOT hierarchical method for the top 6 species of 15 common species in Taiwan sea. It evaluates the posterior probability of the testing samples and eliminates less confident results. A substantially lower false positive rate is achieved.

4 Computation time of Video and Image Processing modules

The video and image processing modules analyse the video data to detect and recognise the fish in the video footage. The video streams are saved as 10 minute video clips. In total there are 528624 video clips. For clarity reasons, we will state both the numbers of videos given in Deliverable 5.4 (which are measured at May 27th 2013) and the new numbers currently in the database.

The fish detection component finished processing all the videos in the database. The fish recognition component finished processing almost half of all the videos, currently because we are focussing on the more promising class of videos, thanks to the new video classification algorithm (i.e. videos without blur or encoding errors). Afterwards we will continue processing the rest of the videos. In the previous report, we mentioned that there are in total 623472 clips. There are multiple clips where we have both low resolution and high resolution videos of the same scene, which we no longer take into account (e.g., in Table 10).

Measurements	Total	Fish Detection (May 2013)	Fish Recognition (May 2013)	Fish Detection (Oct 2013)	Fish Recognition (Oct 2013)
Processed Videos	528624	70784 (13.4%)	67468 (12.8%)	530660 (100%)	271722 (51.4%)
Processed Videos (class normal)	75806			75806 (100%)	74906 (98.81%)
Fish	124m			124m	53m
Fish Detections	1445m			1445m	654m
Speed		40 min (std 83 min)	175 min (std 381 min)	12 min (std 12 min)	160 min (std 246 min)

Table 10: In this table, the comparison between the previous deliverables status of the processing and current status of the processing is stated, where large difference in the number of processed videos are shown. In the previous report, we already stated that lots of videos are blurred or have video encoding effects. Currently, we have method to filter out these kind of videos allowing us to focus first on the more promising class of normal videos

The average computation time to run the fish detection and recognition is improved, as shown in Table 10. There are still however large deviation which can be explained by the fact that there are different video resolutions present in the database. Improvements in both fish detection and recognition components have allowed us to process faster the higher resolution videos in the database.

The stability of the current fish detection and recognition components can also be checked in our system. In deliverable 5.4, we mentioned that 9.3% and 6.7% of the process executions gave errors for respectively the fish detection and recognition component. Given that this system has been running for over half a year, these can be network failures, disk failures, errors in programs, etc. The reported errors of the detection and recognition components were also due to the various tests of software in some cases (i.e. we voluntarily interrupted the jobs for testing purposes, for instance to check the robustness of the system). The current failure rate is 1.2% and 5.5% for respectively the fish detection and recognition components, meaning that these components are very stable. Often, running on the same video again fixes the issues with execution errors, since the workflow component is able to detect and rerun components in error.

5 Workflow Performance and Evaluation

The workflow component is the mediator between the user interface and the video and image processing (VIP) modules that are installed on high performance computing (HPC) platforms. Whenever a user query cannot be fully answered by a lookup to the F4K database by the user interface team, the F4K workflow component is consulted. Among the queries that the workflow is able to answer are the following:

1. Perform fish detection and tracking task on a set of videos from a user-defined start date, end date and camera location(s), with or without user-defined detection algorithms.
2. Perform fish species recognition task on a set of videos from a user-defined start date, end date and camera location(s), with or without user-defined species recognition algorithms.
3. Estimate how long a query from type 1. an 2. above will take to perform on the available HPC facilities.

4. Abort a user query that is currently being executed.

The workflow component is fully integrated on NCHC's HPC platform and can distribute workflow jobs onto the Windrider supercomputer and VM cluster. The main elements of the workflow are the i) workflow engine and ii) workflow monitor. The workflow engine makes its own decision on which platform to use for running a job, which detection or recognition algorithm to select (unless provided by the user), what priority level to set and any dependencies between jobs. It composes abstract and concrete workflows and sends them for execution on the HPC platform. The workflow monitor and oversees the execution of the workflow jobs so that the overall success rate of completed jobs is maximised. In particular, it deals with various scenarios that have been introduced in Deliverable 5.4.

The improvements in the workflow component concerns the workflow monitoring. Previously, failed jobs (those that exited abnormally) were rerun up to 2 times. Now, to save time and resources, we rerun them only once, and targeted them to be rerun on the most stable resource of the platform. In Windrider, this is the *monos01* queue, while on the VM cluster, it is the *gad202* queue. Another improvement concerns job dependencies. The former dependency scenario was that a fish species recognition job on a video has to wait for a fish detection and tracking job on the same video. Previously, if the fish detection job failed, its corresponding fish species recognition job did not get removed from the queue, causing the queue to have jobs that were going to wait indefinitely. Currently, we ensure that the dependent job is treated appropriately, as with the job that it is depending on. If the fish detection job is rerun, the fish species recognition job is also rerun. If the fish detection job is abandoned, then the fish species recognition job is also abandoned. Finally we are also able to track and handle a new type of error, the "queue error". Sometimes a queue (or resource) is down and can cause the job that is executing on it to be delayed or to fail. On Windrider, LSF takes a bit of time to resubmit an executing job to a more reliable queue. However, this delay can take up to more than a few minutes. On the VM cluster, a queue error would cause a job to fail. On both occasions, the workflow treats them as failed jobs and resubmits them to a more reliable queue.

With these changes, we have recalculated the average execution times of software components based on the latest stable VIP modules, as shown in Table 11. These execution times more accurately reflect the execution times of the software components.

We have also rerun workflow queries using the latest stable modules on different quality of data in order to observe its overall performance. The datasets taken were from Typhoon Tembin (24th August 2012) and Diving Experiment (conducted on 5th February 2013). The rationale for selecting these events was the former contained many low quality videos and the latter had a higher percentage of good quality videos. We calculated the percentages of software errors and missing videos on these datasets and compared them to a set of 8,556 videos from the rest of the F4K collection. Figures 10 show the percentage of these errors as identified by the workflow system.

We have been considering a workflow Quality of Resilience (QoR) metric for evaluation, which is the usage of the likelihood of a workflow instance failing. The QoR aims at assisting with the optimisation of the workflow's performance. We introduced QoR in Deliverable 5.4 which spun out as a result of collaborating with University of Cardiff, U.K. and University of Zaragoza, Spain. This collaboration is on-going and initial results are described in [15]. We hope that we can build a data mining model from past experiences to improve the workflow's performance.

Table 11: The performance metrics of the stable software components for fish detection and tracking (IDs 135, 141 and 142) and fish species recognition (IDs 128 and 135) in F4K. *Note* : * denotes default component.

Component ID	Avg. Execution Time(s)	Avg. Queuing Time(s)	Max. Execution Time(s)	Min. Execution Time(s)	Avg. DB Wait Time(s)
128 recognition	8796 (∼2.4hrs)	6164 (∼1.7hrs)	355381 (∼4days)	15	68
135* detection	734 (∼12mins)	90 (∼1.5mins)	19604 (∼5.4hrs)	0	93
136* recognition	9902 (∼2.75hrs)	42655 (∼11.5hrs)	344113 (∼4hrs)	16	32
141 detection	892 (∼14.7mins)	31460 (∼8.7hrs)	2845 (∼47mins)	10	4
142 detection	11336 (∼3.15hrs)	53205 (∼4.8hrs)	28107 (∼7.8hrs)	180	11

6 User interface performance and evaluation

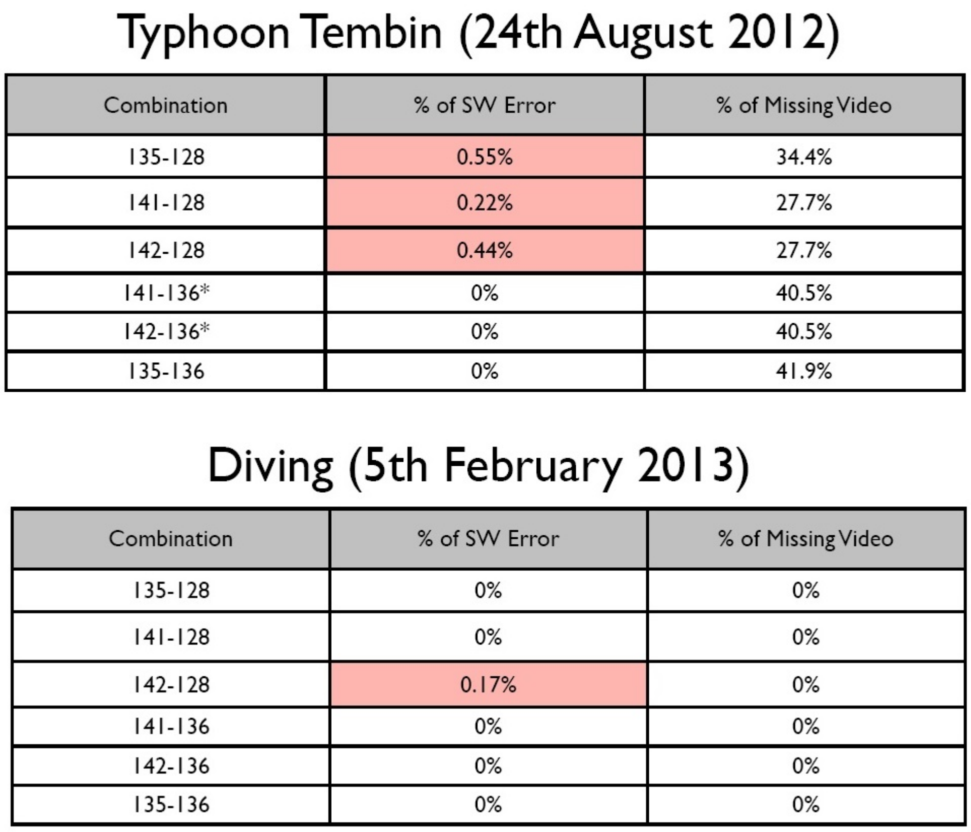
6.1 Functionalities

The User Interface (UI), also called *public query interface*, addresses the needs for i) visualizing the Fish4knowledge data, and ii) evaluating the uncertainties that impact the scientific data analysis. The functionalities are described in detail in deliverable D6.6, and are summarized below.

Present the Fish4Knowledge system - Users are provided with explanations about the video analysis components (in the *Video Analysis* tab), the data extracted from the video (in the *Raw Data* tab), and the functionalites to explore the data (in the *Home* page).

Explore the video collection - Users are provided with a video browser (in the *Video* tab) allowing the exploration of videos with specific criteria (e.g., from specific time and location, with specific image quality, or species).

Explore the data extracted from the videos - Users are provided with a flexible data visualization system (in the *Visualization* tab). It provides visualizations of fish counts, species counts, video counts or average fish counts per video (e.g., to compensate for varying numbers of video samples). Users can define what kind of graph they want to visualize (i.e., line chart, stacked graph, or boxplot). Users can adapt the graph by specifying what the axes should represent. The Y axis can represent fish counts, species counts, video counts or fish counts per video. At the moment, the X axis can represent time periods, it could later include dimensions like locations, species or certainty scores. In the case of stacked graph, users can specify on



Typhoon Tembin (24th August 2012)

Combination	% of SW Error	% of Missing Video
135-128	0.55%	34.4%
141-128	0.22%	27.7%
142-128	0.44%	27.7%
141-136*	0%	40.5%
142-136*	0%	40.5%
135-136	0%	41.9%

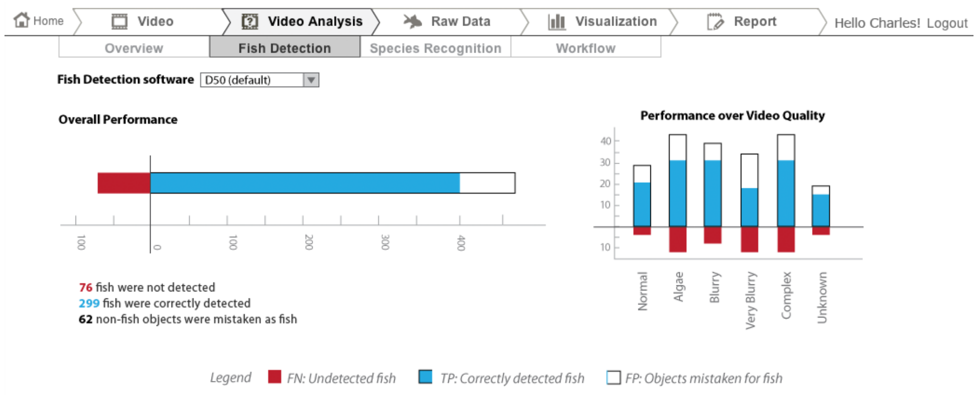
Diving (5th February 2013)

Combination	% of SW Error	% of Missing Video
135-128	0%	0%
141-128	0%	0%
142-128	0.17%	0%
141-136	0%	0%
142-136	0%	0%
135-136	0%	0%

Figure 10: Percentage of software error and missing videos in a dataset with high percentage of bad quality videos (typhoon) versus a dataset with high percentage of good quality videos (diving).

which dimension the graph is decomposed (e.g., at the moment fish counts can be decomposed be camera, or per species). Similarly, in the case of boxplots, users can specify on which dimension the data is subsampled (e.g., to display what is the variation of fish counts over the weeks, or over the hours of the day). Users are provided with widgets for selecting the dataset of interest (e.g., fish appearing at specific time or location, fish from specific species). The widgets open and close on user demand, so that the UI is not cluttered, and users can organize their working environment. The widgets also contain histograms displaying the data corresponding to each option of the filter. For instance, users could be provided with overviews of fish distributions over years, cameras, species, or any other dimension handled by a filter widget.

Compare trends observed in the data - Users are provided with simple means to compare potential trends observed in the data. The *Report* tab allows users to collect and annotate several visualizations that were specified in the *Visualization* tab. These sets of visualizations can be downloaded to be stored or shared with others, and uploaded on the *Report* tab to be re-visualized.



Fish Detection software D50 (default)

76 fish were not detected
299 fish were correctly detected
62 non-fish objects were mistaken as fish

Figure 11: **The Video Analysis Tab - Fish Detection sub-tab** provides simplified visualizations of ground-truth based evaluation of the Fish Detection component. Evaluations are provided for each video quality.

6.2 Implemented refinements

The UI was updated with refinements reflecting the user feedback collected during 34 interviews of marine researchers, and reported in deliverable D2.4. Additional refinements concern technical issues (latency).

Understandable data provenance information - Users required to be provided with understandable, e.g., simplified, evaluations of the video analysis components. We designed and implemented such visualizations as shown in Fig. 11.

Comprehensive data provenance information - Users required to be provided with additional information about video quality (e.g., how many videos are *Blurred*, what are the fish counts for *Blurred* videos). We implemented a widget dedicated to the selection of video quality. It allows to visualize datasets extracted from videos with specific image quality. Users can also browse videos of specific quality. Further, the evaluation of Fish Detection performance is provided for each video quality (Fig. 11). Thus users can evaluate the potential uncertainties due to video quality.

High-level information needs - Users required functionalities to visualize the number of species occurring in the dataset of interest. This functionality is accessible in the UI. Users can specify that the Y axis of the graph must represent species counts (i.e., *species richness*). Users also required to access a description of each fish species. The UI provides an access to species descriptions on fishbase.org, which is a well-accepted tool within the marine biology community.

Connection to the workflow - The UI includes functionalities for users to request specific video analyses to be performed. Users can choose the set of videos to be processed, and the versions of the components that must process the videos. Letting users choose the exact version of the components is useful for replication and comparison purposes. Users can verify if the observed results are the same with the selected component versions. Further, users can control that the same components' versions are used to process the dataset under analysis. Using undifferentiated mix of component versions can create biases in the observed results.

Reduction of latency - The processing time of common user queries could be reduced by i) optimizing the database views and indexing, ii) simplifying the aggregation of data from common queries into visualization, and iii) locating the web server and the database server on the same continent, instead of Asia (Taiwan) and Europe (The Netherlands). We addressed these 3 points, and reduced the latency by improving the schema and indexes of the *summary tables* (described in D2.5), and by installing the UI on web servers localized in Taiwan and in The Netherlands, close to potential users. The current query execution times are not compared with those reported in D5.4, because the amount of data is now much larger than at the time D5.4 was written.

Unaddressed requirements - Users requested information about the *rates of duplicates* (e.g., how likely is it that a single fish is detected several times). Some species may produce more duplicates, depending on their swimming behaviour. For instance, sedentary fish staying at the same coral coral head are likely to produce more duplicates than fish swimming straight forward (e.g., Dascyllus reticulatus). This can create important biases, and over-estimation of some species. Evaluating rates of duplicates was not performed because insufficient data was collected, and no appropriate method was established in the User Interface. A preliminary experiment was conducted. Divers and cameras collected observations of the same ecosystems, for the same period of time. The number of observations was limited, and the observed fields of view were not those of the Fish4Knowledge cameras. Further research is needed to establish a method to normalize fish counts depending on the chances of observing duplicates. In particular, the method must take into account the field of view of each cameras (e.g., sedentary fish occur more in close-ups on a coral head, than in views on open sea).

Users requested to use *calendar and lunar month* to filter datasets of interest. They also required to use widely-used *traditional analyses for biodiversity research*. These functionalities were not implemented, since we focused our resources on refinements related to uncertainty issues. Further, we assumed that these data analyses are specific to particular research interests. The UI cannot address a large variety of specific user needs. However, the UI allow users to perform a preliminary exploration of the data, prior to proceeding to further analyses within their dedicated working environment (e.g., to apply the specific metrics used for biodiversity research).

6.3 Future work

Further research is needed to explore uncertainty issues and their potential solution, as well as the usability of the visualization system. The most important work on usability concerns:

- The evaluation of the *Visualization* tab. A study was performed on simplified data analysis tasks, where participants were given precise instructions on which trends to evaluate or compare. This study is reported in [1]. Further experiments would concern i) the usage of the UI for a complete data analysis task, letting users freely explore the datasets, ii) the comparison with user behavior when performing the same tasks with a more traditional

UI, i.e., that is not using filter widgets with histograms, or that is not offering adaptable axes of graph (letting users choose what is represented by each axis).

- The evaluation of the simplified visualization of video analysis performance (Fig. 11). We assume that our new design is more understandable and intuitive. Further research can be done to verify this assumption.
- The evaluation of user needs for tutorials about exploring the video data. Further research can be done to determine what didactic means can support users with understanding and getting familiar with i) the provided dataset (e.g., the real entities it represent, the characteristics used to describe them), ii) the uncertainties that are contained in the dataset, iii) the exploration of data within the F4K interface.

The most important work on uncertainty issues concern:

- The evaluation of the rates of duplication (single fish detected several times).
- The evaluation of the risks of confusing species. For instance, *species m and n* may have high chances to be confused with *species x*. And only *species m* may be likely to be confused with *species y*.
- The differentiation of random errors (noise) from systematic errors (biases). For instance, the processing of *Blurred* videos may systematically contain different errors for different species, compared to *Normal* videos.
- The correction of potential biases through the normalization of fish counts. For this purpose, the application of a statistical technique called *logistic regression* is currently being studied.

7 Conclusions

This deliverable shows that the project has achieved a fully working system, where the fish detection is run on all videos in the database and the fish recognition processed half (51.4%) of all the videos, where we are currently processing the remainder, prioritizing videos that do not have “blur” or “encoding error” problems. The user interface is able to show the processed information to the users. The user interface is also more understandable and comprehensive with respect to the provenance of the information, and more visualisation options allow user to explore high level information about the data. The connection between the user interface and the workflow has been developed allowing marine biologists to control which video processing modules to use when analysing the videos. The user interface now also provides the marine biologists with more information about the potential errors of the system (e.g., for each video quality), although further research in this subject is needed (e.g, noise and biases).

7.1 Possible Future Work

In this project, we discovered how difficult it is to interpret big data which contains different kinds of uncertainty due to recording setup, video processing, etc. We have observed that groundtruth data plays an important role, where marine biologists would like to understand on what basis the software works and what kind of mistakes and biases in the result they can expect. Groundtruth evaluations can partially answer those questions, showing marine biologists graphs of how the detection and recognition perform on the groundtruth data. We also discovered that showing videos where fish detection and recognition data is plotted into the video gives marine biologists a better understanding of the system. Groundtruth evaluation can also be used in theory to correct for biases in automated fish counts, with the assumption that the groundtruth data is representative of the complete dataset. However, computing such fish count normalization makes the system very complex for marine biologists and further research is needed both on the statistical assumptions as in methods for representation. In our research, we also observed that groundtruth data for learning classifiers might not be a good representation of the entire dataset. The best example is that the entire dataset has a couple of resident species which are dominating the set of available images. For training the classifiers, the number of rare species images need to be relatively large. We focused our efforts on annotating rare fish species, biasing the distribution of the training set. Another interesting direction is linking the groundtruth with the workflow, allowing the workflow to more dynamically change the settings of the video processing software based on this information. In this case, the workflow can look to which class (i.e "blur", "encoding error", "normal") a video belongs and based on this information look at performance on groundtruth video of similar classes and decide which video processing software to select based on accuracy (computed using the groundtruth) and speed (determined from earlier other runs). Other factors than the current evaluation of species recognition accuracy can also be taken into account and presented to marine biologists, allowing them to make better decisions on running the video processing components.

References

[1] Elvira Arslanova. Video analysis software for scientific use: impact of data provenance information on user trust, acceptance and information processing. Technical report.

[2] Olivier Barnich and Marc Van Droogenbroeck. ViBe: a universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, June 2011.

[3] Gary R Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface, 1998.

[4] M. A. T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[5] Phoenix X Huang, Bastiaan J Boom, and RB Fisher. Underwater live fish recognition using balance-guaranteed optimized tree. In *Proc. of ACCV*, 2012.

[6] M Isard and A Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1), 1998.

[7] Isaak Kavasidis, Simone Palazzo, RobertoDi Salvo, Daniela Giordano, and Concetto Spampinato. An innovative web-based collaborative platform for video annotation. *Multimedia Tools and Applications*, pages 1–20, 2013.

[8] Liyuan Li, Weimin Huang, Irene Y H Gu, and Qi Tian. Foreground object detection from videos containing complex background. *Proceedings of the eleventh ACM international conference on Multimedia MULTIMEDIA 03*, 03:2, 2003.

[9] Shengcai Liao, Guoying Zhao, V. Kellokumpu, M. Pietikainen, and S.Z. Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1301–1306, 2010.

[10] Shengcai Liao, Guoying Zhao, Vili Kellokumpu, Matti Pietikainen, and Stan Z Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:1301–1306, 2012.

[11] M. Narayana, A. Hanson, and E. Learned-Miller. Background modeling using adaptive pixelwise kernel variances in a hybrid feature space. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2104–2111, 2012.

[12] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[13] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances In Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[14] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance Tracking using Model Update Based on Lie Algebra. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[15] O. F. Rana, R. Tolosana-Calasanz, G. Nadarajan, C. L. Yang, and Y. H. Chen-Burger and. Analysing Quality of Resilience in Fish4Knowledge Video Analysis Workflows. In *International Workshop on Clouds and (eScience) Applications Management - CloudAM*, 2013.

[16] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1778–1792, 2005.

[17] Concetto Spampinato, Simone Palazzo, Daniela Giordano, F P Lin, and Y T Lin. Covariance-based fish tracking in real-life underwater environment. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2012.

[18] Concetto Spampinato, Simone Palazzo, and Isaak Kavasidis. A texton-based kernel density estimation approach for background modeling under extreme conditions. *To appear on Computer Vision and Image Understanding*.

[19] C Stauffer and W E L Grimson. Adaptive background mixture models for real-time tracking. *Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Cat No PR00149*, 2(c):246–252, 1999.

[20] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequièvre. A benchmark dataset for foreground/background extraction. In *ACCV 2012 / Background Models Challenge*, 2012.

[21] Y.-C. F. Wang and D. Casasent. A support vector hierarchical method for multi-class classification and rejection. In *International Joint Conference on Neural Networks, 2009. IJCNN 2009*, pages 3281–3288, 2009.

[22] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 51–56, 1996.

[23] Jian Yao and J-M Odobez. Multi-layer background subtraction based on color and texture. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[24] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.