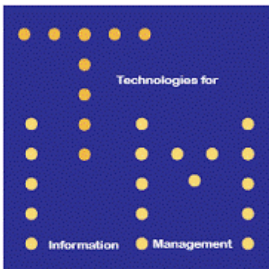

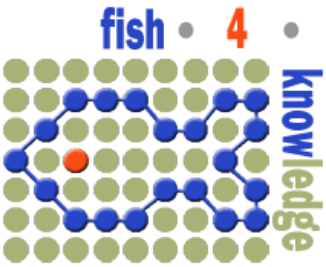

Fish4Knowledge Deliverable 5.4

Experimental evaluation report 1

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

Abstract: In the first experimental evaluation report, we describe the methodology and data used to evaluate the different components in our system. In this report, we evaluated both the separate components (fish detection and tracking, fish recognition, workflow and the user interface) and also look at the evaluation of the entire system. Currently, a prototype system is running and has already processed 67468 clips of 10 minutes videos (10% of total amount of videos) allowing us to show some premilinary analyses of the system to marine biologists. However, the marine biologists only see the user interface, so during the evaluation, awareness of the underlying system was neccesary to obtain feedback. Based on the current system, marine biologists especially indicated that more clarity is neccesary on the performance metrics used in the video and image processing and how this influences the results. From an integration perspective, some final connections between the user interface and workflow components have to be developed so the user can also interact with the VIP components directly.

Deliverable due: 30 Month

1 Introduction

In the Fish4Knowledge project, we have developed a fully working prototype system at the end of the second year of the project. In this system, all of the components developed by the different research groups can communicate with each other using mostly the initial system interface defined by Deliverable 5.1 and 5.2 (only small details changed). This design has proven to be very flexible during the ongoing development, where new versions of individual software components are easy to include into the system without other researchers having to adapt their software components in major ways.
This report will discuss the ongoing efforts to evaluate the individual software components and the overall system. Having a first prototype system available helps basically all research groups with the evaluation of their individual components. In a lot of cases, for the evaluation, research groups also depend on the results of other components in the system. For instance, the workflow is dependent on being able to run the Video and Image Processing (VIP) components in order to adjust their strategies according to the specific properties of those software components. Also, the user interface relies on the data that is made available by the video and image processing components and discussions with users based on fiction data is much harder than discussion on real and maybe interesting results.
To evaluate the entire system, we have to first evaluate the different software components of the system separately. The evaluation of the individual components are based on the criteria that the developers of these components are using for their own evaluation. Most of the evaluation measures of the individual components are very specific to that component, because they all have a different task within the overall system. Still the performance of the individual components also partly reflects on the overall system. For instance, if the fish recognition is not able to recognise species accurately, then the results in the interface might not be accurate, making good evaluation of the different individual components important. Besides looking at the individual components, it is useful to look at how well the entire system is performing. In this case, we looked at two aspects of the entire system, namely how well everything is integrated and what the feedback of the users is on the system. In a lot of cases, the feedback of the users comes from users looking at the interface, however, lots of issues are not related to the user interface but the underlying methodology and software. This is why the user interface is an ideal tool to obtain feedback on the entire system, but not all improvements are at the interface level, where we also need the integration to support the needs of the interface.
This deliverable is organized as following: In Section 2 the state of the prototype system will be discussed. Afterwards, the evaluation of the different components are separately described for fish detection and tracking component in Section 3, fish recognition in Section 4, computation time and stability of VIP in Section 5, workflow in Section 6 and the User Interface in Section 7. In Section 8, conclusions will be given based on the user feedback and the current performance of the system, allowing us to suggest improvements for the last stage of the system development.

2 State of System

In this section, we will discussed the current state of the prototype system developed by the Fish4Knowledge project. Although, we already have an entire working system, there are still several things that need to be developed for the final system. Also, during the course of the

project, new proposed ideas have emerged that are not fully integrated and need more attention. However, currently we have for video and image processing multiple software components that can perform either fish detection and tracking or fish recognition. For fish detection, we have several different background subtraction methods and different fish tracking methods which can be used for this task, because different kinds of methods work better on different kinds of scene.Given the output of the fish detection and tracking, we have a old version of the fish recognition that is able to recognize 10 species, while our newer version recognizes 15 species. For the fish recognition, different preprocessing filters can be used to obtain more accurate fish contours depending on the computation time and necessary accuracy. Currently, a simple bulk processing workflow is used to compute the backlog of video data with both the default fish detection and recognition software components. These components save their data into a database, which is accessible by the user interface. Because the amount of data in the database is massive, summary tables are being used so that the user interface can deal with the amount of data more quickly. Currently, the user interface is able to show statistical information about the processed video data, where we already have years worth of data stored in the database.
In this system, the following connections need further development to improve the system. First, the current bulk processing workflow needs to be replaced by a workflow that can perform bulk processing as well as response to user requests allowing users to run different versions of the software to verify for instance hypotheses. Here a connection between user interface and workflow is also necessary. Second, the integration of how to deal with uncertainty in the system can be improved, where at the moment we already have different levels of score given decisions made by the fish detection and recognition components. However, relating these scores to values that are useful to the end users is difficult, especially because the end-user are also not familiar with these kind of new systems. These are still issues that we are working on, but these are not challenging from a integration perspective, because they mainly involve creating an updated version of an existing component that inputs better quality data in the database.

3 Fish Detection and Tracking

The evaluation phase for the fish detection and tracking algorithms aims at assessing the following properties:

- capability of the detection algorithms to identify the number of fish which are present in a single frame (Section 3.1);
- accuracy of the detection algorithms in the extraction of the contour for each fish (Section 3.1);
- capability of the tracking algorithms to follow a fish throughout a video (Section 3.2);
- accuracy of the total count of fish provided by the tracking algorithms for a whole video (Section 3.2).

3.1 Fish detection

The evaluation of the fish detection perfomance was carried out by testing the performance, first, of the background modeling approaches which identify the foreground pixels that are then

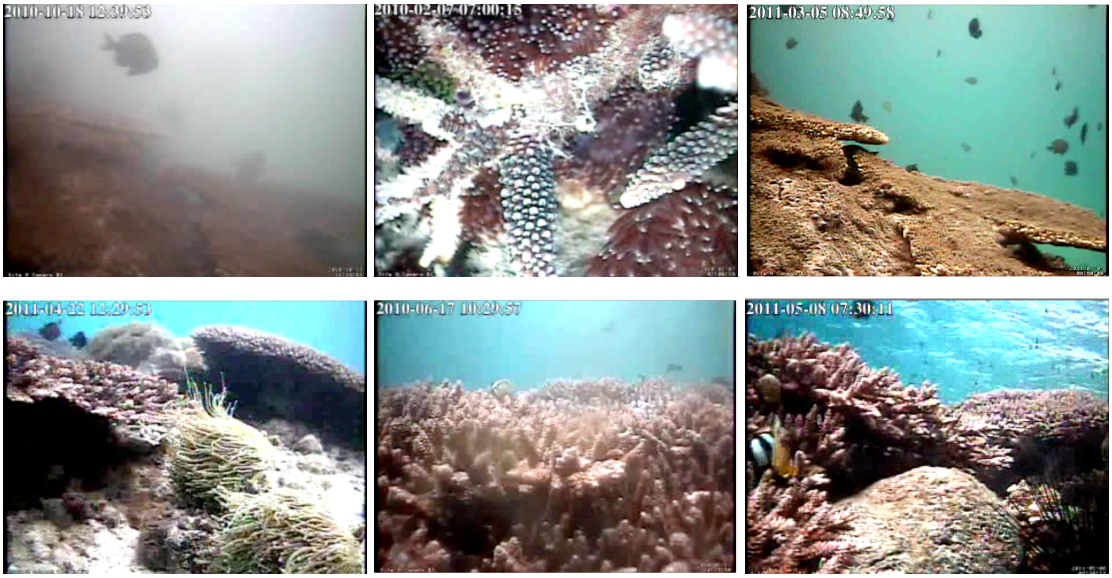

Figure 1: Underwater Video Dataset. From top-left to bottom-right: 1) Blurred, 2) Complex Background Texture, 3) Crowded, 4) Dynamic Background, 5) Luminosity Change, 6) Camouflage Foreground Object

grouped together to form objects (fish), and, then, of the post-processing module which, instead, aims at filtering out the false positives due to errors of the previous step.

In particular, to test our background modeling approaches we used a set of 14 underwater videos (spatial resolution ranging from 320×240 to 640×480) taken from the Fish4Knowledge repository. These videos were categorized into seven different classes to account for the scene variability. The video classes (see Fig. 1) are: "Blurred" (smoothed and low constrast images), "Complex Background Texture" (background featuring complex textures), "Crowded" (lot 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).

The ground truth (consisted of about 20 labeled images per video) on this dataset was hand-labeled using the tool in [8] and it is available together with the videos at `http://f4k-db.ing.unict.it/datasets.php`.

In detail, we compared the results of our method (an adaption to the underwater domain of the VIBE [1] approach) with some state-of-the-art approaches (mainly kernel density estimators) on the same underwater dataset. In order to avoid any implementation bias in the performance evaluation, we used only methods for which the original code was available[1]. More specifically, the following methods were used for comparison:

- P-Finder [17] which models the background with only one single Gaussian *pdf* (Gaussian):
- Two methods that exploit mixture of Gaussians, namely, the original Gaussian Mixture

[1]Most of the methods are available at `https://code.google.com/p/bgslibrary/`. The code of the remaining methods were made available by the authors and reference to the code can be found in the corresponding papers.

Video Class	P-finder	GMM	$ZGMM$	$EIGEN$	$ML-BKG$	$KDE-RGB$	$VIBE$
Blurred	75.26	83.30	77.84	81.71	70.26	**92.56**	85.13
Complex Background Texture	75.63	66.95	75.94	74.78	**83.67**	87.53	74.17
Crowded	71.22	**85.17**	74.41	73.87	79.81	82.46	84.64
Dynamic Background	51.03	62.04	64.30	71.48	**77.51**	59.13	67.01
Hybrid	74.64	62.71	75.50	80.69	72.20	**85.69**	79.75
Luminosity Change	48.10	63.08	59.07	70.41	**82.66**	72.06	70.37
Camouflage Foreground Object	72.43	66.25	70.03	70.20	73.51	54.14	**76.30**
Average	66.90	69.92	71.01	74.73	**77.08**	76.22	76.76

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

Model by Stauffer and Grimson [16] (*GMM*) and its improvement by Zivkovic (*ZGMM*) [19];

- The Eigenbackground (*EIGEN*) Subtraction method [9],
- Two non-parametric kernel density estimation approaches: the Sheikh's method [12] (*KDE-RGB*), which uses colour features for modeling the background, and the Multi-Layer background model (*ML-BKG*) by Yao in [18], which, instead, employs also texture features computed via Local Binary Patterns.
- *VIBE* [1] that models the background through actual pixel color values instead of using a predefined *pdf* shape;

The performance of the different methods are reported as F-Measure values (at the algorithms' best operating points) and illustrated in Table 1, which shows that combining colour and texture features (as in *ML-BKG*) enhanced the object detection performance also in complex scenarios where targets and background had similar texture features.
Of course, the increment in accuracy of the *ML-BKG* was achieved at the expenses of efficiency; in fact, the average number of frames (size 320×240) processed per second for *VIBE* and *ML-BKG* were, respectively, 200 frames/sec and 20 frames/sec with a C++ implementation running on a PC powered by an Intel i7 3.4 Ghz CPU and 16GB RAM. The other information that can be derived from Table 1 is that methods relying on a *pdf* with a predefined form (e.g. P-Finder, GMM, ZGMM) are not suitable to deal with complex scenes, where, instead, non-parametric methods perform much better.

3.1.1 Detection post-processing

Fish detections are extracted from the binary motion masks (i.e. the output of the background modeling process) by searching for connected regions of foreground pixels. However, due to the complexity of the monitored environment (lighting changes, sunlight gleaming, plant movements), many false alarms, i.e. image regions mistakenly identified as fish, may be detected. In order to avoid such mistakes, a detection post-processing stage [14] has been introduced to filter out objects which do not show motion and appearance traits typical to fish.

A Naive Bayes classifier has been used to discriminate between *good* and *bad* detections; the training and test sets were built by manually labeling 852 samples, equally divided between images representing single fish (as positive samples) and background portions (as negative

samples). Each sample consisted in the frame at time t, in which the object appears, the corresponding binary mask and the frame at time $t + 1$ (used for the computation of optical flow descriptors). The precision and recall scores achieved by the module are respectively 89.4% and 97.3% (overall misclassification rate of 6.8%).

3.2 Fish tracking

The evaluation of tracking algorithms was performed using both a typical ground-truth-based approach (where the tracker's trajectories were compared to the hand-labeled ones) and a self-evaluation approach, which did not require the availability of ground-truth data and consisted in analysing the likelihood that a given trajectory was correct, based on a motion and appearance consistency analysis.

3.2.1 Ground-truth evaluation

The ground truth for fish tracking was built by hand-labeling a set of 5 videos (duration: 10 each minutes, resolution: 320×240, frame rate: 5 fps), for a total of 1854 trajectories and 27212 single detections (resulting in an average trajectory length of about 15 detections). The annotation process involved both the manual labeling of fish contours and their associations across frames.

The evaluation approach compared the ground-truth trajectories to the ones obtained by the tracking algorithms, in order to produce the following set of quantitative indicators, describing the accuracy of the algorithms from several points of view:

- 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 2 shows the results we obtained with the covariance-based tracker [10], the CAMSHIFT tracker [3] (which was used in the only other known application of underwater tracking [13]) and the CONDENSATION [7] tracker. As can be seen, the best results in all three indicators was obtained by the covariance tracker, closely followed by CONDENSATION, whereas CAMSHIFT obtained significantly worse performance.

3.2.2 Self-evaluation

Because ground-truth generation is very tedious and time-consuming (and therefore error-prone), we also sought confirmation for the results shown in the previous paragraph by an alternative

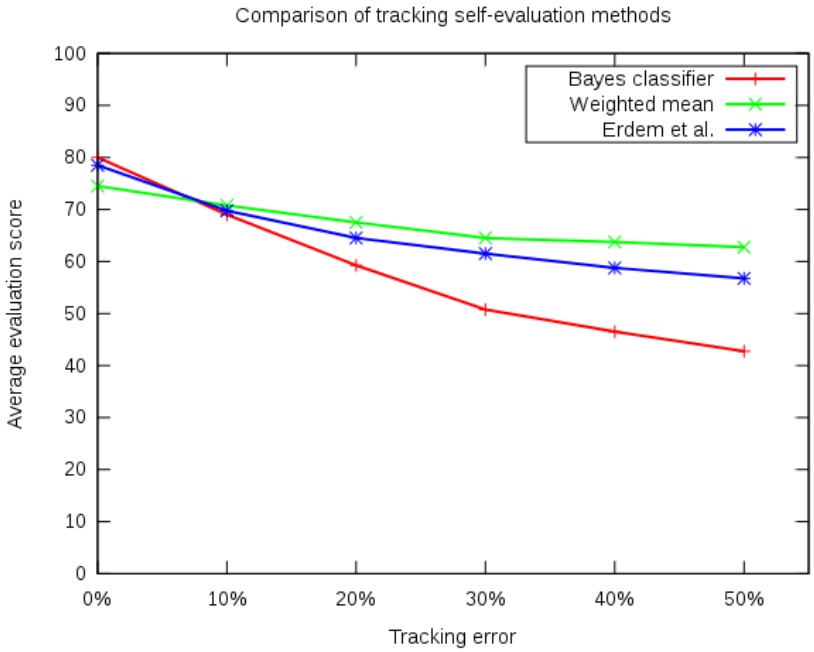

Figure 2: Ability of our tracking self-evaluation approach to reflect the presence of tracking errors, with respect to two similar methods (see text for details).

evaluation approach, consisting of an analysis of the motion and apperance consistency of a trajectory as an indicator for the likelihood that the trajectory is correct.

This analysis assigned a *quality score* to each tracking decision (as defined in the previous paragraph), based on whether the new location of a tracked object was consistent, in terms of motion pattern (in direction and speed) and visual description (shape, area, texture and intensity histograms), with the object's history up to that point. Details can be found in [15]; basically, the algorithm employs a Naive Bayes classifier to merge motion and appearance information features.

The quality score for a given trajectory was computed as the average score assigned to each of the relevant tracking decisions, and the overall evaluation score for the algorithm was computed as the average of all trajectory quality scores.

In order to estimate the accuracy of the self-evaluation approach we designed a test consisting in introducing a certain amount of artificial tracking errors into the ground-truth data (by altering object contours and/or switching or removing associations between objects in different frames) and evaluating how the average quality score varied with respect to the amount of errors introduced. Figure 2 shows the results of this test and a comparison with 1) a variant of the algorithm which computes a weighted mean of the above-mentioned features, and 2) a similar approach described by Erdem *et al.* [4]. It is easy to notice how our approach is able to better reflect the amount of tracking errors introduced in the ground-truth data.

Table 2 shows the results of the self-evaluation approach for the three tracking algorithms under examination. We can see that this method is able to reflect the higher accuracy of the covariance and CONDENSATION trackers with respect to CAMSHIFT, while at the same time showing the similarity between the results obtained by the first two.

3.2.3 Discussion

After this evaluation, the two candidate trackers were the covariance-based one and CONDENSATION, since the former obtained a higher accuracy on the ground-truth evaluation, whereas the latter had a slight advantage in the self-evaluation. In the end, since that advantage was too small to be statistically significant, and because self-evaluation approaches unavoidably suffer an intrinsic uncertainty, the covariance tracker was chosen as default tracker for the historical video processing.

	Covariance tracker	CAMSHIFT	CONDENSATION
CCR	91.3%	83.0%	90.9%
ATM	95.0%	88.2%	93.6%
CDR	96.7%	91.7%	94.5%
Self	80.1%	76.3%	80.2%

Table 2: Comparison between the results obtained by the covariance-based algorithm, CAMSHIFT and CONDENSATION on the ground-truth data and with the self-evaluation approach.

4 Fish Recognition

4.1 BGOT-based hierarchical classification

We apply a hierarchical classification method for fish recognition by using a Balance-Guaranteed Optimized Tree (BGOT) [6]. Firstly, the BGOT algorithm arranges more accurate classifications at a higher level and leaves similar classes to deeper layers. Secondly, it keeps the hierarchical tree balanced to minimize the max-depth and control error accumulation. This method controls the error accumulation during hierarchical classification and, therefore, achieves better performance.

Based on the BGOT tree, we develops two improvements: node rejection and trajectory voting.

The node rejection in Figure 3 algorithm aims at controlling the error accumulation. It adds a ”-1” branch at each node. This branch contains all hidden classes which do not appear in this node. Any fish that is classified as ”-1” will be rejected, and these rejected fish are re-classified by using a flat SVM.

The trajectory voting in Figure 4 is used to minimize the environmental influence. As all fish are freely swimming in a varying illumination environment, the detected fish may have different orientations and appearances. Therefore, the recognition results may vary even for a fish in the same trajectory. The trajectory based voting mechanism is applied after individual classification. It combines the single frame classification results. The trajectory voting method enhances the fish recognition accuracy by exploiting the consistency in labels expected from tracking each fish individually. A voting mechanism (winner-take-all) is then carried out within each group which reduces the misclassification rate.

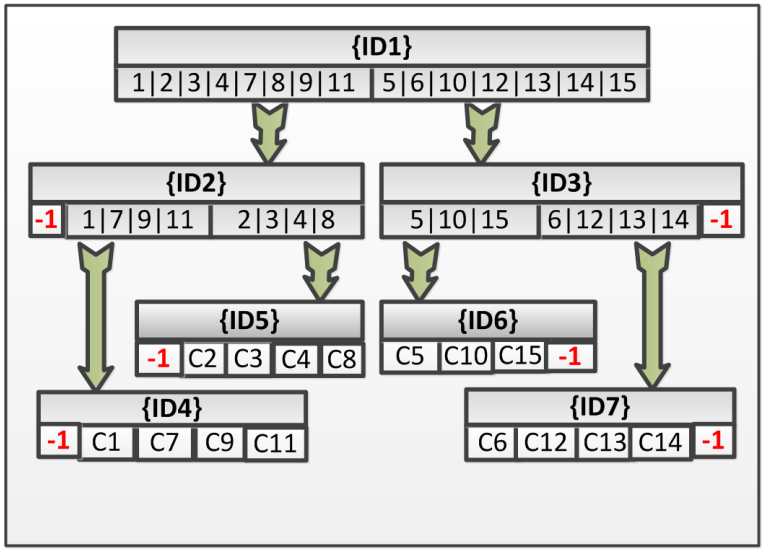

Figure 3: A Balance-Guaranteed Optimized Tree computed from training data is shown, where the leaf nodes contain classifiers that either separate the fish in more subclasses or reject the fish for a particular subnode (shown by the "-1" branch), because it is not similar to the fish species in that particular node. Rejected fish are classified by a flat SVM in this case.

4.2 Groundtruth Dataset

To evaluate fish recognition, lots of groundtruth data is necessary for both training the fish recognition methods and evaluating these methods. Using clustering to support the annotation [2], helps us to produce a large set of around 91894 images where a lot of images are labelled as "bad images", because of occlusions, low resolution, etc. For 28,264 images, we have obtained species labels at the moment, however these numbers are still increasing. Currently, we are training our new fish recognition methods using these dataset, so the experiments and evaluation are still performed on a older subset of 6874 fish images. New efforts are being focussed on finding rare species in this data, ignoring the common species where enough training and testing data is available. Currently, we are annotating a new set of images where we filtered out common species in the dataset.

4.3 Evaluation and analysis

The experiment is based on 6874 fish images with a 5-fold cross validation procedure over 15 species. The training and testing sets are isolated so fish images from the same trajectory sequence are not used during both training and testing. Sequential forward feature selection is applied at each node. Results are listed in Table 3 where the result score is averaged accuracy over all classes rather than over all fish. This is because of the greatly unbalanced class sizes.

Three performance metrics are employed to evaluate the accuracy of the proposed system. The first metric is Average Recall (AR) over all species. It describes on average how many fish are correctly recognized for each species. This score is more important to our experiment because of the imbalance in the classes. The second score is Average Precision (AP) over all species. It is the probability that the classification results are relevant to specified species. The

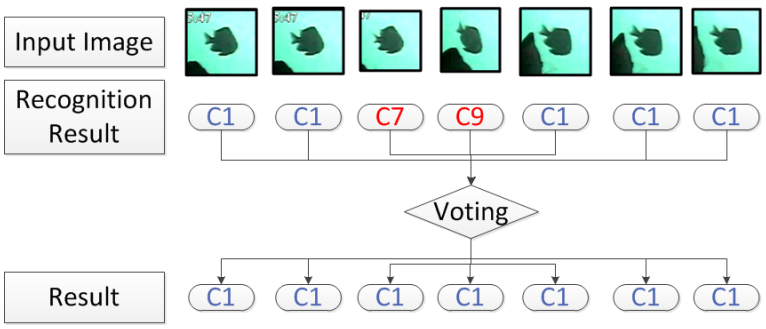

Figure 4: An example of trajectory voting is shown where we use a winner-take-all strategy.

Method	AR (%)	AP (%)	AC (%)
SVM	72.61 ± 6.02	77.63 ± 3.22	93.22
PCA (95%)	72.74 ± 6.39	76.68 ± 3.51	93.09
SVM (fs)	76.71 ± 5.93	81.47 ± 5.27	93.50
taxonomy	77.16 ± 6.29	80.80 ± 6.92	93.76
BGOT method	**85.75 ± 5.64**	**91.30 ± 8.73**	*** 97.21**

Table 3: Fish recognition results. Our proposed result is in **Bold** font. We add the standard deviation of AR/AP/AC over 5 fold cross validation. * means the AC is a significant improvement over other methods at 95% confidence level.

third metric is the accuracy over all samples (Accuracy over Count, AC), which is defined as the proportion of correct classified samples among the whole dataset.

We also have compared our new result with the previous one [6]. We use the same introduced dataset, and choose the top 10 species because the previous component can only recognize these 10 species. The AR score shows that our new component achieves 81.03%, which is 14% higher than the old component (65.63%).

5 Computation time of Video and Image Processing modules

The video and image processing modules allow our system to analyse the data. Currently, this data is saved in 10 minute video clips, however, over the time of the project the resolution of the video clips has improved and different formats have been used. As of May 27th 2013, the fish detection has processed 70784 clips of 10 minutes which is equal to around 983 days of video given the 12 daylight hours we are recording. The fish recognition, which depends on the fish detection component has processed around 67468 clips of 10 minutes (937 days of video). In total, we have however 623472 clips, although there are multiple clips where we have both low resolution and high resolution videos of the same scene. There are also a lot of videos where both fish detection and recognition is very hard due to very blurred recording, coding errors, etc. Currently, filters are being developed to flag these videos, allowing us to focus on the more promising videos first.

The average computation time to run the fish detection and tracking on a single videos is currently 40 minutes with standard deviation of 83 minutes. The large standard deviation is due to the fact that there are different kind of resolutions, where high-resolution videos need much more computation time. To compute the fish recognition, it takes on average 175 minute and the standard deviation is 381 minutes. Notice this large standard deviation is mainly due to the fact that there are several videos containing many more individual fish than other videos and also here, the resolution of the videos has a large influence. In the future, we hope to develop a fish recognition method that is better able to deal with high resolution videos, which is currently a bottle-neck for the existing fish recognition software components.

The stability of the current fish detection and recognition components can also be easily checked in our system. In the case of the fish detection, 9.3% of the fish detection component executions reported an unknown or known error to the system while computing the videos. Given that this system has been running for over half a year, these can be network failures, disk failures, errors in programs, etc. By reporting these errors, videos can be run again to check if the error is something temporary or if it will be a persistent error. The fish recognition has reported a 6.7% fish recognition video error. After inspection, lots of error can be usually resolved by running the software again because resources where unavailable. The reported errors of detection and recognition components may be due to the various tests of software in some cases (i.e. we voluntarily interrupted the jobs for testing purposes to check for instance the robustness of the system). Although improving this is not our top priority, we are still monitoring the error log in case new issues arise.

6 Workflow Performance and Evaluation

A workflow management system is required to process on-demand queries from the user interface and internal batch queries on NCHC's high performance computing platform and to monitor their execution. On-demand queries are those that have to be processed immediately because it originates from the user. These on-demand queries can be one of the following:

1. Detect and track all the fish in a given date range and set of camera locations.
2. Identify all the fish species in a given date range and set of camera locations.
3. Estimate how long a detection or recognition query above will take to execute.
4. Abort a query that is in the middle of being processed.

Internal batch queries are those that are invoked by the workflow management system itself. These are predominantly batch tasks on new video clips that have been collected and recorded in the database by NCHC. It involves running fish detection and tracking (Query 1 above). These batch queries are considered to have low priority so that they do not interfere with user queries if they are spawned at the same time. Batch queries are scheduled to be executed at quiet times, i.e. when on-demand queries are least likely to be processed.

At present the compute environment consists of a cluster of virtual machines (approx. 42 CPUs) and the Windrider supercomputer (96 CPUs). The workflow is able to schedule 300 batch jobs (i.e. process 300 videos) every 24 hours and listens for new on-demand queries

every 10 seconds. It can handle various scenarios on the Windrider facility while development is on-going to have similar handling strategies on the VM cluster. This is because they both use different resource schedulers (LSF on Windrider and Gridengine (SGE) on VM cluster) and different mechanisms are needed to deal with these two schedulers. When the jobs are executing, the workflow monitors them for successful completion, or deals with errors that occur. We define test scenarios to demonstrate the strategies that we have deployed to deal with the jobs dispatched for execution.

6.1 Different scenarios

We have tested various query processing and error handling strategies by simulating scenarios. These queries are triggered using database entries onto the `query_management` table for new queries or updates onto the `job_monitoring` table for existing queries. Real data and stable software modules have been used for all the tests. Some scenarios were difficult to simulate, for example a job waiting for more than one hour (most queues were efficient enough to process jobs within this time frame and we did not have access to a whole queue exclusively to do such tests). In these cases we tweaked the time thresholds to shorter values to demonstrate that the situation can be handled. The first five scenarios were all tested on the Windrider facility. Scenarios 6.1.6 and 6.1.7 are applicable to the VM cluster.

6.1.1 Successful Completion

This is the default scenario that occurs to most jobs scheduled for execution. When a job completes successfully, the scheduler returns with exit code 0 (DONE). The workflow monitoring system catches this and updates the `job_monitoring` and `query_management` tables.

6.1.2 Failed Jobs

This scenario is simulated by running fish recognition tasks using incorrect parameters. The program will thus fail and return with an error exit code. When this is encountered, the workflow monitoring will reschedule the job for execution. A failed job is rerun at most twice. After that it will be abandoned by the system and recorded in the database tables as being "abandonedBySystem".

6.1.3 Job Dependencies

This scenario applies to fish species recognition jobs. Fish species recognition can only be applied when the fish objects have been detected in the videos. Thus a fish recognition module can only be applied to a video when a fish detection module has already processed it. The workflow engine deals with a fish species recognition task as follows:

- If fish detection has been completed on the video, then run fish recognition only.
- If fish detection has not been started, run fish detection and fish recognition with dependency flag between them.
- If fish detection has been started but not completed yet, run fish recognition with dependency flag on the running fish detection job.

6.1.4 Different Priorities

Jobs can have different priority levels when being sent for execution. As stated above, on-demand queries will be tranlated to high priority jobs by the workflow engine, while batch queries are translated to low priority jobs. Higher priority jobs will get precedence for execution over lower priority jobs when being sent to a queue. Two main scenarios have been detected when different priority jobs are queuing and executing in one queue:

- A low priority job has been waiting for too long: they are resubmitted with higher priority.
- A high priority job has been waiting for too long: suspend low priority jobs that are running and resume them after a threshold.

6.1.5 User Aborted Queries

When a query is being executed, the user may decide to give up on it. They can choose to abort the query. Queries aborted by the user will be marked as "abandonedByUser" and its corresponding unfinished jobs will be killed.

6.1.6 Successful Completion on VM

On the VM group, successful jobs run using the gridengine scheduler (SGE) can be monitored and marked as completed by the workflow, same as Scenario 6.1.1 above.

6.1.7 Failed Jobs on VM

Similar to Scenario 6.1.2, failed jobs can be rerun up to twice on the VM cluster using SGE.

6.2 Evaluation Measure and Performance Based on these Measures

In order to evaluate the workflow system implemented, we decided to note the effects on the overall system when the workflow is not used (jobs are send directly to resource scheduler). Table 4 below summarises how each scenario is handled in the presence and absence of the workflow. Additionally it states the possible effects on the system when the workflow is not used.

We have also introduced Quality of Resilience [11] metrics to improve workflow's overall performance a metric that identifies how resilient a given workflow is likely to be prior to its enactment. QoR aims at specifying workflow resilience from three different perspectives: user (QoRU), workflow enactor (QoRE) and resource manager (QoRR). We assume that a user running a workflow is primarily interested in a submit-and-forget mode of operation i.e. where a workflow is submitted to an enactment engine often subject to a number of different constraints, also identified by the user (such as execution time, cost (financial and in terms of resources used) of execution, etc). Whereas significant work in workflow enactment has focused on performance (often measured as the workflow makespan) and the associated Quality of Service (QoS) metrics, limited attention has been given to resilience.

In F4K, we have defined a workflow that takes two tasks, fish detection and tracking (t1) and fish species recognition (t2) in a sequential manner to accomplish a fish species recognition query. For this task, we have determined 4 workflow options for t1 (corresponding to 4 detection

algorithms) and two recognition options for t2. This would yield 8 combinations or workflow variants to be used. The main aim of the QoR analysis is to give a likelihood of failure by using each combination of t1 and t2 options. The likelihood of the workflow to fail when using each combination is calculated based on the performance (execution time) of the components. Progress of this work is contained in [5].

Table 4: How jobs are handled with and without the workflow system in different scenarios

Scenario	System Handling using Workflow	System Handling without Workflow	Possible Effect(s) without using Workflow
Successful Completion	Finished	Finished	All jobs are waiting in the default queue without utilising full system capacity
Failed Job	Rerun at most twice	Exit directly	The failed job will not be detected until the user checks the status manually
Job Dependency	With error handling for failed jobs	Without error handling	The failed dependent job will be taken care by the system
Different Priority: High priority job waiting too long	Suspend low priority jobs that are running	Job keeps waitting in the queue	If the queue is packed with low priority jobs, high demand user query will be held for a long time
Different Priority: Low priority job waiting too long	Resubmit with higher priority	Job keeps waitting in the queue	If the queue is packed with high priority jobs, low priority jobs can be starving in the queue

In summary, we can conclude that when the system does not make use of the workflow, suitable resources and queues are not being selected. Jobs that fail are not rerun and in extreme cases some jobs can starve. All these factors affect the overall system performance.

Table 5: Statistics of average execution times and waiting times (in minutes) for fish detection (80) and fish recognition (52) on Windrider for the month March 2013.

component_id	average_execution _time (sec)	average_waiting _time (sec)	num_videos _processed	maximum _time (sec)	minimum _time (sec)
52	11,248 (>3 hrs)	561 (>9 mins)	4,457	469,603 (>130 hrs)	0
80	8,000 (>2 hrs)	32,176 (<9 hrs)	2,180	77,271 (>21 hrs)	2

Table 5 shows the statistics of the jobs executed on Windrider using two main components for fish detection and fish species recognition in March 2013. These jobs were executed without using the workflow engine and monitoring system. It can be seen that the average waiting time, and maximum execution time should be reduced to enhance the overall performance of the F4K system. This will be our aim when we migrate the workflow from testing phase to live production phase.

7 User interface performance and evaluation

7.1 Answering user queries

The current User Interface (UI) supports flexible data visualizations over the entire dataset. The data visualization system allow users to access various kinds of graph, which are all based on similar database queries (the *common queries*). Table 6 summarizes the common database queries used for the current UI demo. The response time of these queries is given in Table 6. The response time for database queries was calculated using the NCHC database, and using extensive *where* clauses since they imply the longest response times. The overall response time, from the user query to the update of the UI, is much more important than the response time for database queries. It impedes the usability of the Fish4Knowledge system since users experience noticeable latency when requesting a new kind of graph, or when refining the dataset filters (e.g., it can be around 10s). The latency comes from the cumulated response time for database query, the aggregation of data from common queries, the transfer of data from the database located in Taiwan to the browser of the end user (e.g., for the biologists located in the Netherlands). The reduction of latency relies on i) the efficiency of the database views and indexing (e.g., the *summary tables* available for each camera), ii) the optimization of the data processing for integration in interactive visualizations (e.g., the aggregation of data from common queries), iii) the relative geographical location of all the servers involved in the data processing, from the database to the end user.

7.2 User feedback

We conducted a series of 34 interviews of researchers within the coral reef biology community in Taiwan and in the Netherlands. The collected feedback gives directions for addressing further user information needs, and trust and uncertainty issues. The user information needs expressed in the early user studies can be addressed by the current UI design. But they can be further addressed by including additional data visualizations. We will study and prioritize the potential design refinements and integration of new UI features. The most important user feedback mentioned by users concern:

- **Comprehensible and relevant data provenance information**: the information related to the performance of the video analysis components is difficult to understand since users are not familiar with computer vision techniques. We observed that users tend to overlook the technical details that can bias their analysis. Further work can be done to refine the report on the certainty scores attributed to each image processing steps, and on the ROC-like evaluation of the video processing. Further, users expressed demands for other technical information regarding uncertainty issues:
  - The quality of the video (e.g., fuzziness, murkiness) that can for instance bias the seasonal pattern observed, since seasonal events can influence video quality;
  - The rate of duplicates of single fish in fish counts. Some species may produce more duplicates than other species. This is an important bias for studying the relative abundance of each species (e.g., species composition);
  - The performance of the video analysis components for various sets of video, for instance more errors may occur with murky videos.

Table 6: Common queries used for the current F4K demo. These queries are used for the visualization of raw fish counts and normalized fish counts (i.e., the average number of fish per video).

	Fish count		
Q1	Per year	select year(date) as year, count(fish_id) from summary_camera_38 group by year order by year	1.590s
Q2	Per week	select week(date) as week, count(fish_id) from summary_camera_38 group by week order by week	1.736s
Q3	Per hour	select hour(date) as hour, count(fish_id) from summary_camera_38 group by hour order by hour	1.678s
Q4	Per camera	select count(fish_id) from summary_camera_38	1.648
Q5	Per species	select species_id, count(fish_id) from summary_camera_38 group by species_id order by species_id	1.656s
Q6	Per software version	select det_component_id, rec_component_id, count(fish_id) from summary_camera_38 group by det_component_id, rec_component_id order by det_component_id, rec_component_id	1.717s
Q7	Per certainty score	select truncate(rec_certainty, 1) as rec_cert, count(fish_id) from summary_camera_38 group by rec_cert order by rec_cert	1.657s
	Video count		
Q8	Per year	select year(date_time) as year, count(distinct v.video_id) from video v, (select v1.video_id from processed_videos v1, processed_videos v2 where ((v1.component_id=50 and v2.component_id=52)) and v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id group by year	0.300s
Q9	Per week	select week(date_time) as week, count(distinct v.video_id) from video v, (select v1.video_id from processed_videos v1, processed_videos v2 where ((v1.component_id=50 and v2.component_id=52)) and v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id group by week	0.303s
Q10	Per hour	select hour(date_time) as hour, count(distinct v.video_id) from video v, (select v1.video_id from processed_videos v1, processed_videos v2 where ((v1.component_id=50 and v2.component_id=52)) and v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id group by hour	0.301s
Q11	Per camera	select v.camera_id as camera, count(distinct v.video_id) from video v, (select v1.video_id from processed_videos v1, processed_videos v2 where ((v1.component_id=50 and v2.component_id=52)) and v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id group by camera	0.306s
Q12	Per software version	select v3.det_component_id as det_component_id, v3.rec_component_id as rec_component_id, count(distinct v.video_id) from video v, (select v1.component_id as det_component_id, v2.component_id as rec_component_id, v1.video_id from processed_videos v1, processed_videos v2 where v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id group by det_component_id, rec_component_id order by det_component_id, rec_component_id	4.241s
Q13	Total	select count(distinct v.video_id) from video v, (select v1.video_id from processed_videos v1, processed_videos v2 where ((v1.component_id=50 and v2.component_id=52)) and v2.status in (”completed”, ”completed - workflow error”) and v1.video_id = v2.video_id) v3 where v.video_id = v3.video_id	0.306s

- **High-level information needs**: a number of additional visualizations and UI features were suggested by users, such as: the integration of a calendar and of lunar month for filtering datasets of interest, the calculation of the number of species that occur in the dataset of interest (*species richness*), the usage of the traditional data analyses widely used for biodiversity research, access to detailed description of fish species (e.g., image of the species, link to fishbase.org).

8 Conclusions

8.1 Refinements for the integration

In this deliverable, we show that most of the system is fully integrated and the system has already processed lots of videos (67468 clips) as promised. The user interface team is able to read this processed information and show statistics to the users. During interviews with marine biologists, a better understanding of the underlying computer vision is necessary for potential usage. In the integration, we have to focus with the user interface team on improving the user understanding by presenting measures that are comprehensible and useful to marine biologists. In the system, the user should also have control over which video and image processing components are used when analysing certain videos. In this case, a connection between the workflow and the user interface still needs to be developed to support users allowing them to perform new analyses on the data or verify parts of the data using multiple video and image processing components.

8.2 Refinements suggested by biologists

The interviews of researchers within the coral reef biology community provided valuable insights for understanding the potential usage of our tool, and more generally, for understanding the acceptance of video analysis tools by the marine biology community. Video analysis tools are relatively recent in this community. No well-accepted data analysis framework has been set up for the usage of video data for marine biology research. Particularly, the evaluation of the video analysis algorithms is a technical information of main concern for the acceptance of video analysis tools. Biologists need to evaluate the potential errors (e.g, noise and biases) contained in the data. However, the types of evaluation that are well-accepted by the image processing community are not easy to understand by the marine biology community. And biologists may require additional information, such as the rate of duplicates in fish counts (e.g., the chance of counting single fish several times), and detailed evaluations under specific conditions (e.g., performance for murky videos only).

During this first end-to-end evaluation of the system, users also suggested interesting refinements of the interactive features of the UI. Our future work consists of eliciting the most important user requirements to implement within the duration of the project. We will particularly focus on the features that allow biologists to use video analysis data for scientific research (e.g., the provenance information, and the support of uncertainty issues).

References

[1] 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.

[2] B. J. Boom, P. X. Huang, J. He, and R. B. Fisher. Supporting ground-truth annotation of image datasets using clustering. In *21st Int. Conf. on Pattern Recognition (ICPR)*, 2012.

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

[4] C.E. Erdem, A. Murat Tekalp, and B. Sankur. Metrics for performance evaluation of video object segmentation and tracking without ground truth. *Proceedings of Internation Conference on Image Processing*, 2:69–72, 2001.

[5] Y.-H. Chen-Burger R. Tolosana-Calasanz O. Rana G. Nadarajan, C.-L. Yang. Analysing quality of resilience in fish4knowledge video analysis workflows. *CloudAM 2013 (submitted)*.

[6] P. Huang, B. Boom, and R. Fisher. Underwater live fish recognition using a balance-guaranteed optimized tree. In *Asian Conference on Computer Vision*, 2012.

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

[8] 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.

[9] 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.

[10] 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.

[11] O. Rana J. Banares D. Talia R. Tolosana-Calasanz, M. Lackovic. Characterizing quality of resilience in scientific workflows. *WORKS'11*, pages 117–126, 2011.

[12] 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.

[13] C Spampinato, Y H Chen-Burger, G Nadarajan, and R B Fisher. Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos. In *3rd International Conference on Computer Vision Theory and Applications, VISAPP 2008*, pages 514–519, 2008.

[14] Concetto Spampinato and Simone Palazzo. Enhancing Object Detection Performance by Integrating Motion Objectness and Perceptual Organization. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR*, pages 3640–3643, 2012.

[15] Concetto Spampinato and Simone Palazzo. Evaluation of Tracking Algorithm Performance without Ground-Truth Data. In *IEEE International Conference on Image Processing, to appear*, 2012.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

