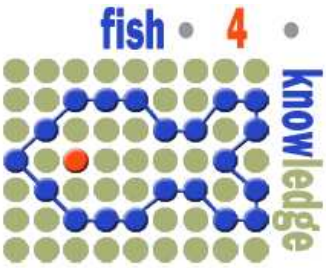


Fish4Knowledge Deliverable D1.3

Fish Recognition and Clustering

Principal Author: Phoenix X. Huang, Bastiaan J. Boom, Robert B. Fisher
Contributors: UEDIN
Dissemination: PU

Abstract: This document describes the fish recognition and clustering methods, which makes use of the fish detection and fish description described in respectively Deliverable D1.1 and D1.2. The document is divided into two parts: fish recognition and fish clustering. In both parts, a small literature study is performed looking at related work. Afterwards the methodology that is used in this project is explained and experiments are performed to show that this methodology works on the obtained dataset (see also Deliverable D2.3). At the moment, the fish recognition methodology is already being used in our first prototype, where it classified around 18 million fish images. The fish clustering methodology has been actively used for annotation where around 300 thousand fish are currently clustered to speed up the fish annotation and groundtruth production process.

Deliverable due: 24 Month

Figure 1: Embedded camera in Fish4Knowledge

1 Introduction

1.1 Why We Want To Recognize Fish?

Human activities have altered the fish distribution around the world. Measuring the number of fish by recognizing the species and their number is valuable for applications such as fisheries management, commercial purposes such as fish farms and shops, and ecological benefits such as evaluating the impact of water related constructions [10]. Statistics about the specific species distribution of fish besides an aggregate count of aquatic animals can assist biologists resolving issues ranging from food availability to predator-prey relationships [40].

1.2 Recent Underwater Surveillance Approaches

Traditionally, marine biologists have employed many tools to examine the appearance and quantities of fish. For example, they cast nets to catch and recognize fish in the ocean. They also dive to observe underwater, using photography [7]. Moreover, they combine net casting with acoustic (sonar) [6]. Nowadays, much more convenient tools are employed, such as hand-held video filming devices [46]. There are two main disadvantages using this equipment. Firstly, these activities disturb fish swimming and habits, and thus giving rise to abnormal situations. This drawback is apparent: the fish are sensitive to their surrounding environment. Secondly, the small amount of acquisition data could not meet the demands for extensive underwater animal analysis, and the recorded data also could be considered as sampling at a very low rate, which leads to a Picket Fence Effect and omits valuable information. To resolve these issues, some researchers implement automatic analysis by using a Digital Signal Processor (DSP) chipset with camera onboard [13]. It is cost-effective and easy to program (using standard C code). The DSP calculates image processing algorithms and records data to its flash memory. A more popular and practical equipment is Remotely Operated Vehicle (ROV) [3, 18] with standard PC computation [43, 52]. This equipment obtains video from mid water and produces high-resolution images of different fish species. The use of an ROV has achieved great success in collecting such data. They generate huge amounts of video containing animals from underwater cameras [46]. However, these techniques have their own shortcomings. A DSP chip with an embedded program cannot perform rapid calculation and an ROV can only stay underwater for a limited time. In the Fish4Knowledge project, embedded video cameras such as in Figure 1

are used to record underwater animals (including insects, fish, *etc.*) at the Third Taiwanese Power Station as well as three other locations, and observe fish presence and habits at different times [35]. The Fish4Knowledge project investigates methods for capture, storage, analysis and query of multiple videos. The project goal is to analyze large amounts of data using a combination of computer vision, semantic web, database storage and query and work flow methods. Figure 2 shows a video acquisition system that is deployed at the HouBiHu station.

Figure 2: Surveillance System in Fish4Knowledge

1.3 Why Using Automatic Image Processing?

Limitation of manpower Nowadays, underwater videos are mostly analyzed by biologists [47], but it is an exhausting procedure. The difficulties are mainly two-fold.

- The huge amount of data. As a camera produces 2×10^{12} bytes data in a year at Fish4Knowledge, it may take 15 years for a marine biologist to analyze, recognize and label fishes in these videos. In the whole project, 10 cameras have been recording for the last six years, which entail about 900 years' manpower to process this huge database. It is sensible to employ some automatic image processing methodologies to help marine biologists analyze them, as the task of video processing is monotonous and complex.

- Image distortion in the underwater environment. When light projects through the water, it changes light's physical attributes and affects underwater recording and image quality. Firstly, impurities in water limit the range of light and water absorbs light. As a result, distant structures in the video are blurred. Secondly, water changes light's path which is called scattering. This phenomenon affects relative positions of objects in video. Thirdly, in the general computer vision model, light consists of three components: red, green and blue. As these components have different wave lengths, they have distinct distortion

models. It has been observed that the blue light has the highest range in the water, which makes blue a more dominant component in color distribution than normal situations. How the underwater distortion model influences the video recording has been described in [44].

Normally, there are two ways to comprehend underwater image processing [43].

- The first way considers this phenomenon as an image restoration problem and tries to use a degradation model. This method aims at recovering the original image from the degraded one. The main disadvantage of this model is about the parameters. As this method demands strict conditions, mass parameters (*e.g.* attenuation and diffusion coefficients [43]) are essential to accomplishing the computation. Another important parameter required is the depth of an object in the scene [38].

- The second way pays attention to image enhancement and it uses qualitative standards to produce a more visually useful image [21, 38]. This method is beneficial in that it does not rely on any physical model for image formation, and therefore is usually easier to implement and faster in performance than the former method.

Based on these two approaches, researchers have achieved many results in the field of underwater fish analysis. [45] used stereo images to get the distance information of fish. However, as suggested in [45], the underwater environment makes it difficult to find a solution. Variability in water clarity and scatter particles can play an important role to affect image quality. Properly triangulating and measuring objects [20,37] are also required in stereo video and laser scanning systems to extract accurate information, such as length/height and girth, or estimate the overall biomass of fish species in the recorded area. In a pertinent publication, [24] used stereo video to chart the 3-dimensional position and movements of underwater animals (dolphins in this case). Their goal, which achieved some success, was to extract complex positional information from a video image clip with less manual effort [30]. They also discussed their approach of using a photograph system that does not require the use of lasers or light grids. As discussed above, Fish4Knowledge uses computer vision based automatic methodologies for underwater fish processing. Nadarajan *et al.* in [35] proposed an integrated work flow system that aims at helping marine biologists annotate fish in underwater videos. The component diagram [4] is displayed in Figure 3. An image processing ontology is combined with work flow executions, and each task has been designed as an algorithm component. This project focuses on the marine biological video data, and the technology can also be applicable to other general purposes, e.g. marine food farm investigation, public environment surveillance, etc. The research result should be an example of co-operation between computer science and the marine biologist.

2 Fish Recognition

2.1 Introduction

Live fish recognition in the open sea has been investigated by [29, 37, 41, 50] for commercial and environmental applications like fish farming and meteorological monitoring. The detected fish are in 3D positions and against coral and sand as well as the open sea. Statistics about the specific oceanic fish species distribution besides an aggregate count of aquatic animals can assist biologists resolving issues ranging from food availability to predator-prey relationships

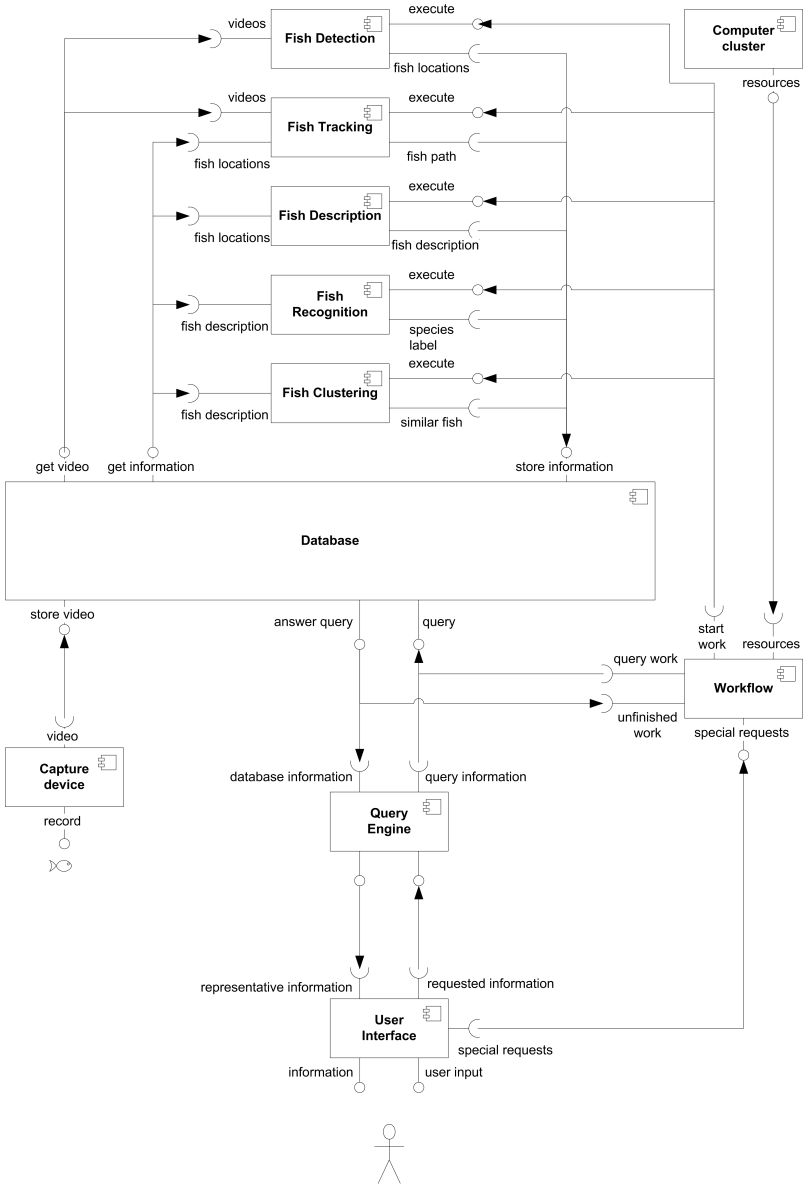


Figure 3: Fish4Knowledge UML component diagram overview [4]

[40, 55]. However, the recognition task is fundamentally challenging because fish can move freely and illumination levels change frequently in such environments [48, 51]. As a result, this task remains an outstanding research problem. Prior research is mainly restricted to constrained environments (*e.g.*, fish tanks [29], conveyor belts [49]). Strachan et al. [50] achieves the scores of 73%, 63% and 90%, respectively, on three types of fish. C. Spampinato et al. [47] classifies 360 images of ten different species and achieves an average accuracy of about 92%. R. Larsen et al. [26] classify three fish species and achieve a recognition rate of 76%. In contrast, this paper investigates novel techniques to perform effective live fish recognition in an unrestricted natural environment.

2.1.1 Related work

SVM method. The fish recognition task is seen as an application of multi-class classification, which has become an important and interesting research area since the influence of machine learning theory. Over the last decade, SVM [9] has shown impressive accuracy on the multi-class classification task because of its maximum-margin advantages. However, SVM is originally designed for a binary classification task. Therefore, to enable multi-class classification, several mechanisms, such as one-vs-one and one-vs-rest, have been developed. This kind of multi-class classifier could be considered as a flat classifier because it classifies all classes at the same time [8] and omits the inter-class correlations. A shortcoming of the flat classifier is that it uses the same features to classify all classes without considering that some classes have certain similarities and can be better separated by some customized features.

Hierarchical classification tree method. To overcome the problem of flat classifier, one possible solution is to integrate a domain knowledge database with the flat classifier and construct a tree to organize all classes hierarchically [11]. This strategy is called hierarchical classification which inherits from the divide and conquer tactic. Essentially, it uses a hierarchical classification procedure where a customized classifier is trained with specific features at each level [19].

Hierarchical classification has several noticeable advantages. Firstly, it divides all classes into certain subsets and leaves similar classes for a later stage. This strategy balances the load of any single node. Secondly, unlike the flat classifier choosing a feature set based on the average accuracy over all classes, the hierarchical method applies a customized set of features to classify specific classes. As a result, it achieves better performance on similar classes than the flat classifier. Thirdly, the hierarchical solution exploits the correlations between classes and finds similar groupings. This is especially useful with a large number of categories [11]. Hierarchical structures are popular in document and image categorization. Mathis [33] organizes documents hierarchically by making use of the correlations between topical subjects. Deng *et.al.* [12] introduced a new dataset called ImageNet where a large scale hierarchical ontology of images are constructed based on the WordNet knowledge. However, these approaches use pre-defined hierarchical structures without considering how to construct a more accurate tree based on given classes.

Nonetheless, the hierarchical structure has a critical disadvantage called error accumulation. Each level of the hierarchical tree may have some classification errors. These errors are accumulated into deeper layers and reduce the average accuracy of the final result.

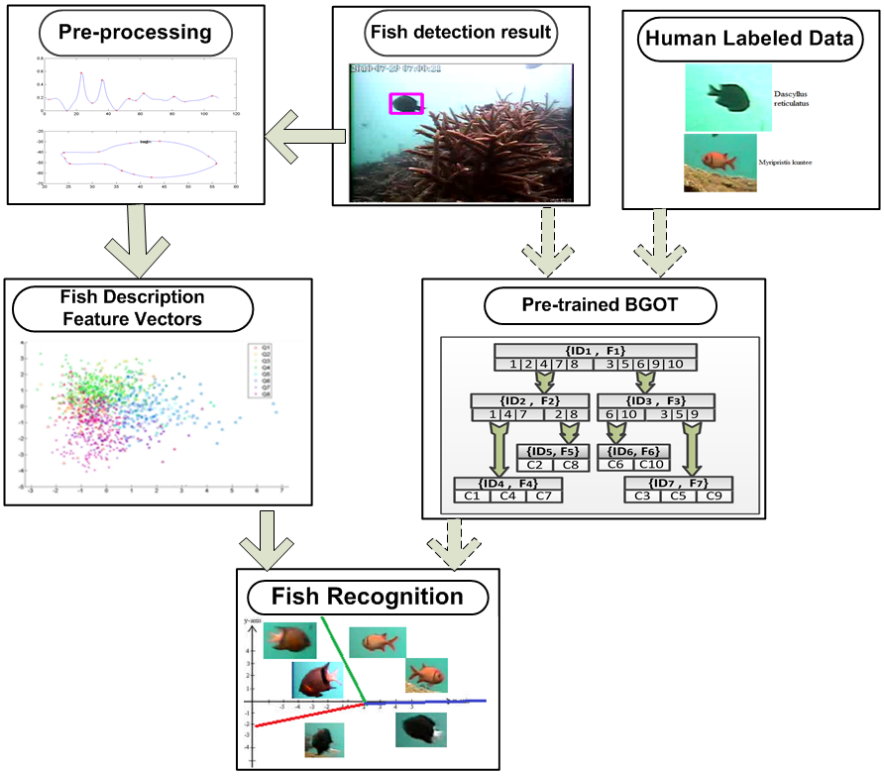


Figure 4: The framework of our BGOT-based hierarchical classification system. The work flow of dotted arrows shows the training procedure and the solid arrows indicate the recognition procedure.

2.1.2 The framework

In this paper, we propose a novel method to recognize fish in an unrestricted natural environment from underwater videos. We use the Balance-Guaranteed Optimized Tree (BGOT) to help resolve the error accumulation issue and make use of the inner-class similarities among fish species. The framework is illustrated in Fig 4.

In this paper we propose a hierarchical classification approach for live fish recognition. Furthermore, we use a heuristic method to construct an automatically generated BGOT and the proposed method is evaluated on a live fish dataset. The algorithm itself is presented in Section 2.2, including the mathematical explanation of hierarchical classification, a set of heuristics which help construct the hierarchical tree. In Section 2.2, we compared the proposed BGOT tree to an Ada-boost [14] method and a flat SVM [9] on a fish image set [36].

2.2 Hierarchical classification approach

Given a set of samples $\{x_i\}_{i=1}^n$, the feature vector $f_i = \{f_{i,1}, ..., f_{i,m}\}$ denotes the m feature values for sample x_i. Let $\{y_i\}_{i=1}^n$ indicate the class label of x_i, and $y_i \in \{1, ..., c\}$ where c is the number of classes. Our aim is to construct a classifier h which uses the feature f_i as input to predict the class label $\tilde{y}_i = h(f_i)$ that maximizes the classification accuracy.

A hierarchical classifier approach h_{hier} is designed as a structured node set. Fundamentally, a node is defined as a triple: $\text{Node}_t = \{\text{ID}_t, \tilde{\text{F}}_t, \hat{\text{C}}_t\}$, where ID_t is a unique node number,

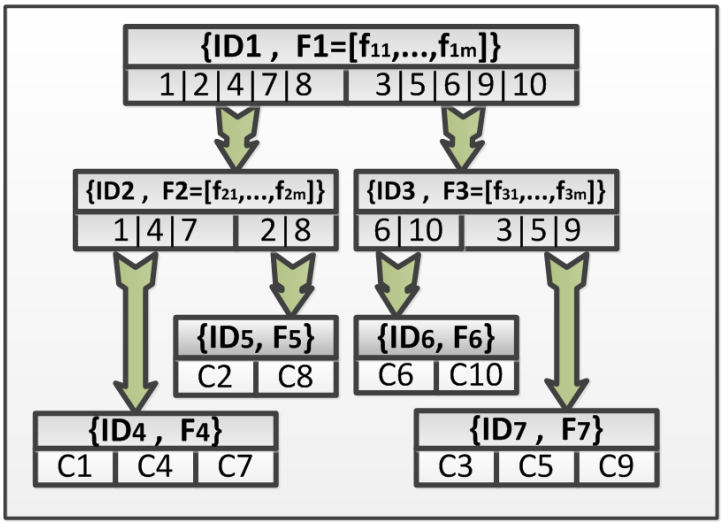


Figure 5: Automatically generated tree (BGOT), the hierarchical example tree of 10 classes ($C1, ..., C_{10}$).

$\tilde{\mathrm{F}}_t \subset \{f_1, ..., f_m\}$ is a feature subset chosen by a feature selection procedure that is found to be effective for classifying $\hat{\mathrm{C}}_t$, which is a subset of classes and their groups. We only consider binary splits at the upper levels so each node has at most two groups. All samples that are classified as the same group will be transmitted into the same child node for later processing. An example with 10 classes is demonstrated in Figure 5, where the ID_t and $\tilde{\mathrm{F}}_t$ are illustrated in each node and $\hat{\mathrm{C}}_t$ is described as local groups.

2.2.1 Heuristic method

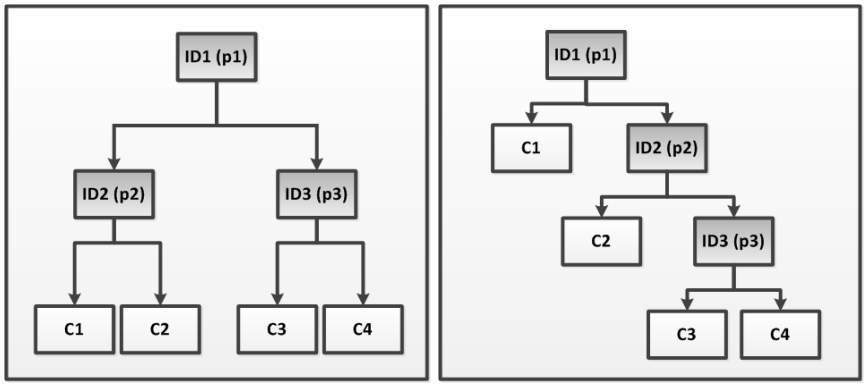
We proposed two heuristics for how to organize a single classifier and construct a hierarchical tree with higher accuracy.

1. Arrange more accurate classifications at a higher level and leave similar classes to deeper layers.

2. Keep the hierarchical tree balanced to minimize the max-depth and control error accumulation.

Rule 1 recommends how to assign the single classifiers to a hierarchical tree. We consider the balanced tree T_b in Figure 6(a) with sample number n_i. This tree has 4 classes $\{c_1, c_2, c_3, c_4\}$ and each single classifier has a different accuracy $\{p_1, p_2, p_3\}$. The average accuracy is calculated as $p_1 * \frac{1}{2}(p_2 + p_3)$ assuming all classes have equal magnitude. The best accuracy is achieved by assigning the most accurate classifier to node ID_1. Generally, the result of a balanced hierarchical tree of N nodes has depth $log_2 N$ and average accuracy:

$$P_b = \prod_{i=1}^{log_2 N} \widetilde{P}_i = \prod_{i=1}^{log_2 N} \frac{1}{2^{(i-1)}} \sum_{s=2^{(i-1)}}^{2^i-1} p_s \quad (1)$$



(a) Balanced tree T_b (b) Unbalanced tree T_u

Figure 6: Examples of hierarchical trees.

where p_s is the accuracy of node s and $\widetilde{P}_i$ is the average accuracy of all nodes in layer i. The hierarchical tree achieves better accuracy if we choose the more accurate classifiers at higher layers which equates to assigning these nodes a higher weight. In the future, we will think more about how to construct the hierarchical tree if classes are not at equal size.

Rule 2 is explained by comparing two sample trees: a balanced tree T_b and an unbalanced tree T_u. These examples are shown in Figure 6. Let us assume each class has the same number of samples n_i and each classifier has an equal accuracy p. In T_b, each class is classified with an accuracy p^2, while the average accuracy in T_u is $\frac{1}{4}(p + p^2 + 2p^3)$. We can prove that $P_b > P_u$, for $0.5 < p < 1$. To generalize, a balanced tree of N nodes has average accuracy:

$$P_b = p^{log_2 N} \tag{2}$$

and unbalanced accuracy:

$$P_u = \frac{1}{N}(\sum_{i=1}^{N-1} p^i + p^{N-1}) \tag{3}$$

for $0.5 < p < 1$, $P_b > P_u$. Thus a more balanced hierarchical tree with $log_2 N$ depth suppresses error accumulation, and achieves better accuracy than an unbalanced tree.

2.2.2 Algorithm of generating BGOT

The BGOT is based on the two heuristics of the last section: keep the hierarchical tree balanced and optimize the performance by putting more accurate nodes at the top layers. In the fish recognition task, some species of fish are more similar than others and the similarity is summarized from the confusion matrix. We illustrate the algorithm of generating BGOT below:

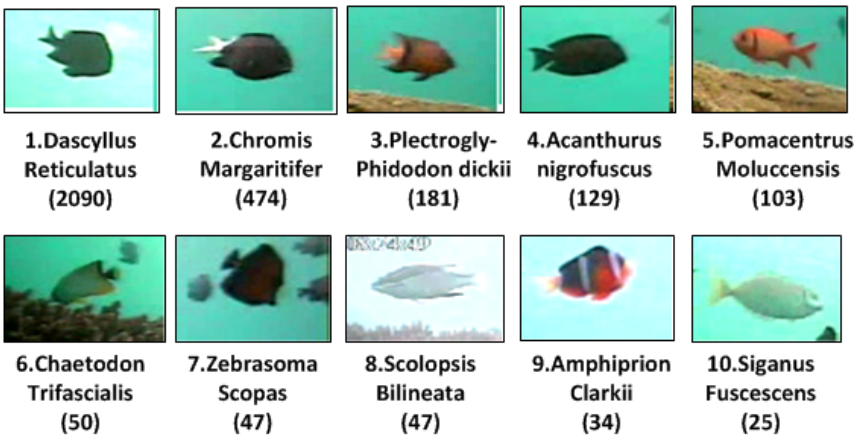


Figure 7: Top 10 species of fish in underwater videos.

Input: class C1 to Cn
begin $c := [C_1, ..., C_n]$
$level := 0$
construct$(c, level)$;
where
proc construct(c, n) $\equiv$
if $n > MAXDEPTH$ then exit fi;
comment: find the best binary split of given classes on whole feature set;
$[cLeft, cRight] := ChooseSplit(c)$;
comment: The ChooseSplit function splits the class set into equal-size subsets;
$featureSet = FeatureSelection(cLeft, cRight)$;
comment: the minimum splitting is set to 3 to limit the max depth;
if $size(cLeft) > 3$ then
construct$(cLeft, n + 1)$
fi;
if $size(cRight) > 3$ then
construct$(cRight, n + 1)$
fi;
end

An example BGOT is shown in Fig 5, where 10 classes are arranged into 3 layers. The first layer splits all classes into two groups: C1, C2, C4, C7, C8 and C3, C5, C6, C9, C10. Then it chooses the feature subset to maximize the average accuracy of these groups. This procedure keeps on until all groups have less than 4 classes.

2.3 Experiment with fish recognition

Our data is acquired from a live fish dataset with 3179 fish images of the 10 different species shown in Figure 7. This figure shows the fish species name and the numbers of images. As can be seen, the data is very imbalanced where the first two species account for 2564 images. The fish detection and tracking software described in [36] is used to obtain the fish images. The fish species are manually labeled by following instructions from marine biologists.

Figure 2.3 shows some hard fish examples: blurred, occlusion by other fish or background

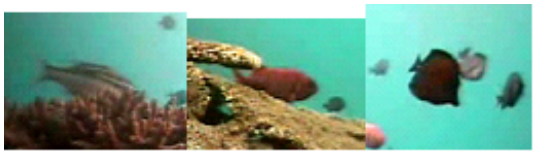


Figure 8: Hard fish examples.

objects, which include coral, the sea flower and open sea.

2.3.1 Feature extraction

After constructing the fish dataset, some pre-processing procedures are undertaken to improve the recognition rate. The Grabcut algorithm [39] is employed to segment fish from the background, which is initialised based on the segmentation obtained from the fish detection. We align the fish images to the same direction before further processing. This procedure is carried out based on a streamline assumption, which assumes that most fish have a smoother head than tail because fish need a more frictional tail (caudal fin) to swim and help them keep balance. We calculate the fish orientation by weighting each contour pixel with its local curvature scale, and we use this algorithm to align all fish horizontally where the head of the fish is located on the right. In order to find the tail side, we smooth the fish boundary with a Gaussian filter to eliminate some noise, and then calculate the curvature of each boundary pixel as following [22, 34]:

$$\kappa(u,\sigma) = \frac{X_u(u,\sigma)Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2))^{\frac{3}{2}}} \tag{4}$$

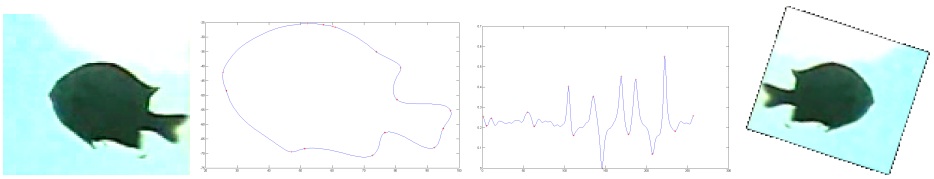
where $X_u(u,\sigma)$/$X_{uu}(u,\sigma)$ and $Y_u(u,\sigma)$/$Y_{uu}(u,\sigma)$ are the first and the second derivative of $X(u,\sigma)$ and $Y(u,\sigma)$, respectively; $X(u,\sigma)$ and $Y(u,\sigma)$ are the convolution result of 1-D Gaussian kernel function $g(u,\sigma)$ with fish boundary coordinates $x(u)$ and $y(u)$. However, the pixel curvature is sensitive to local corners and we normalize it using the logarithm function:

$$\kappa_{normalize} = \begin{cases} log(\kappa) & \text{if } \kappa \geq 1 \\ -log(2-\kappa) & \text{if } \kappa < 1 \end{cases} \tag{5}$$

The fish boundary coordinates are weighted by their local curvature and the vector from the center of mask to the curvature weighed center estimates the tail orientation. A typical fish orientation procedure is illustrated in Figure 9. The fish orientation method achieves 95% accuracy using 1000 manually labeled fish images. Finally, every fish image is divided into four parts (head/tail/top/bottom) according to the relative positions from the fish center.

After this, 66 types of feature are extracted. These features are a combination of color, shape and texture properties in different parts of the fish such as tail/head/top/bottom, as well as the whole fish. We use normalized color histogram in the Red&Green channel and the Hue component in HSV color space. These color features are normalized to minimize the effect of illumination changes. We recompute the range of every bin according to the average distribution over all samples and map them into a 11-bin histogram to take full advantage of all bins:

$$\widetilde{B}_i = \sum_{j=a_i}^{a_{i+1}} B_j \qquad s.t. \qquad a_i = min\{X \in \mathbb{N}^+ \mid \Sigma_{j=1}^{X} \overline{B}_j \geq \frac{i}{11}\} \tag{6}$$



(a) . (b) . (c) . (d) .

Figure 9: Fish orientation demonstration: (a) original fish image; (b) fish boundary after gaussian filter; (c) curvature along fish boundary; (d) oriented fish image.

where $B_j, j \in \{1, ..., 50\}$ is the original color histogram bin, $\overline{B}_j, j \in \{1, ..., 50\}$ is the averaged histogram over all samples and $\widetilde{B}_i, i \in \{1, ..., 11\}$ is the recomputed bin.

In order to describe the fish texture, we calculate the co-occurrence matrix, Fourier descriptor and gabor filter. The grey level co-occurrence matrices describe the co-occurrence frequency of two grey scale pixels at a given distance d [47]:

$$C_{\Delta u,\Delta v}(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{m}\begin{cases}1 & \text{if } I(p,q)=i \text{ and } I(p+\Delta u, q+\Delta v)=j \\ 0, & \text{otherwise}\end{cases} \tag{7}$$

The frequency is calculated for several orientations λ. We compute Contrast, Correlation, Energy, Entropy, Homogeneity, Variance, Inverse Difference Moment, Cluster Shade, Cluster Prominence, Max Probability, Auto correlation, Dissimilarity. These 12 features are useful as they are the most commonly selected features by the feature selection procedure.

Histogram of oriented gradients and Moment Invariants, as well as Affine Moment Invariants, are employed as the shape features. Furthermore, some specific features like tail/head area ratio, tail/body area ratio, *etc.* are also included. All features are normalized by subtracting the mean and dividing by the standard deviation (z-score normalized).

2.3.2 Hierarchical classification

As the SVM is firstly designed for binary classification problems, we introduce a one-vs-one strategy with a voting mechanism to convert the binary SVM into a multi-class classifier [9]. Based on the multi-class classifier, we designed two classifiers (see Figure 5):

1. A flat SVM classifier, which classifies all 10 classes simultaneously, is implemented as a baseline classifier.

2. An automatically generated tree (BGOT) is designed by recursively choosing a binary split which has the best accuracy in given classes. We choose binary splitting to keep the tree balanced.

An Ada-boost method [14], which boosts on individual features, is also implemented as a comparison method.

2.3.3 Results and analysis

The experiment is based on 3179 fish images with a 6-fold cross validation procedure. 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. We then train a customized classifier at each node for specific classes. Results are listed in Table 1 where the AP and AR results are averaged over all classes rather than over all fish. This is because of the greatly unbalanced class sizes.

The accuracy of a classification system is evaluated as Average Recall (AR), Average Precision (AP) and Accuracy over Count (AC). Generally, given True Positive / False Positive / False Negative, the AR is defined as:

$$AR = \frac{1}{c}\sum_{j=1}^{c}(\frac{TruePositive_j}{TruePositive_j + FalseNegative_j}) \tag{8}$$

where c is the number of classes. The second score is Average Precision (AP) over all species. It is the probability that the classification results are relevant to specified species, as shown below:

$$AP = \frac{1}{c}\sum_{j=1}^{c}(\frac{TruePositive_j}{TruePositive_j + FalsePositive_j}) \tag{9}$$

The 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. The AC is calculated as following:

$$AC = \frac{\sum_{j=1}^{c} TruePositive_j}{\sum_{j=1}^{c}(TruePositive_j + FalsePositive_j)} \tag{10}$$

We compare the hierarchical classification against the Ada-boost method (75.3% AR) and flat SVM classifier (86.3% AR). The automatically generated hierarchical tree (BGOT), which chooses the best splitting by exhaustively searching all possible combinations while remaining balanced, achieves an AR of (90.0%). The search procedure takes several hours and a possible improvement is to integrate the hierarchical method with domain knowledge like taxonomy, which helps organize similar species for later processing, instead of exhaustive searching. In the average precision (AP) score, the proposed BGOT method is about 6% better than the baseline SVM method, which are 85.8% and 91.7%, respectively. The Ada-boost method is 76.9% in AP. The AC score of BGOT is tested in a t-test with 95% confidence of significant improvement than the SVM method and Ada-boost method. We calculate the average AC rates at each level in the hierarchy (BGOT): 0.977 (Level 1), 0.9725 (Level 2), 0.950 (Level 3).

Algorithm	AR	AP	AC
Ada-boost	0.753 ± 0.091	0.769 ± 0.092	0.923
Flat SVM	0.863 ± 0.052	0.858 ± 0.061	0.934
BGOT method	0.900 ± 0.042	0.917 ± 0.045	0.950*

Table 1: Fish recognition result. * means significant improvement with 95% confidence.

The individual class recall/precision is shown in Figures 10 and 11. The hierarchical approach achieves a better accuracy than the flat SVM classifier because it arranges the similar species (1,4,7) into the same group and adds fish-tail features to distinguish these species.

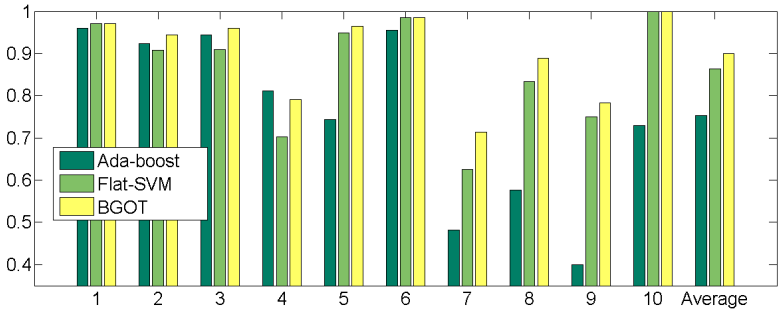


Figure 10: Recall of 10 species. The BGOT method is better than the baseline method.

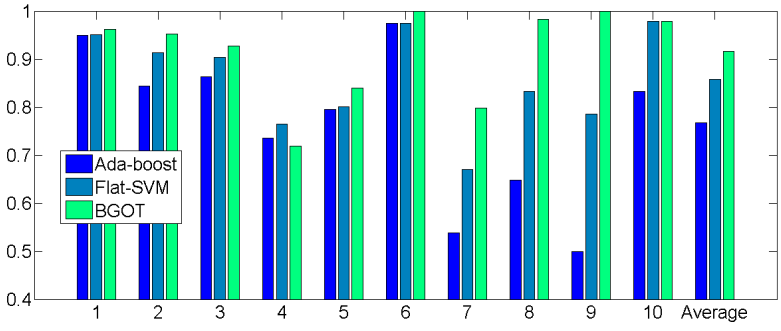


Figure 11: Precision of 10 species. The BGOT method is better than the baseline method (class 4 is an exception, and is discussed in the result section).

3 Fish Clustering

3.1 Introduction

Fish Clustering has as its main purpose to support the fish annotation process. In [5], we show that it is much faster to annotate images by using clustering methods. In [5], we used the clustering method proposed in [15] on the basis of the Information Bottleneck proposed in [17], which will be discussed in the coming section. Based on our experience with the clustering methods [15], we discover that given the amount of data that we would like to annotated, we quickly run into scalability issues.
There are several reasons to perform image retrieval on such a database: Firstly, biologists are interested in searching for rare fish species (where there are not many training images available of such a species, making machine learning methods less suitable). Secondly, the annotation of fish species can be supported by an image retrieval method, where annotators only have to indicate if fish belong to the same species which makes this task faster and easier to perform. Thirdly, given that a large set of images is annotated, this system can also be used for fish recognition, where it searches quickly for similarly annotated fish. For biologists, it is easy to improve the accuracy of this system by adding newly annotated fish images to the set in case of mistakes.
The proposed fish clustering method should be able to deal with low resolution images. One of the problems is that the features of unknown fish that need to be annotated are not always known, making feature selection not an option. Instead, we propose a nearest neighbor search methods that scales for large datasets, is able to deal with low resolution images and is very flexible with comparing unknown features. In section 3.2, a literature study is performed. Section 3.3.1 explains the Information Bottleneck as a distance measure between feature sets. In Section 3.3.2, examples of the feature extraction with the Information Bottleneck are given. Section 3.3.3 describes the method to obtain the binary codes for indexing based on this distance measure. Experiments are performed in Section 3.4 with a large dataset of low resolution fish images which shows the best results in image retrieval with the Information Bottleneck.

3.2 Literature Review

Low resolution images are very common in video surveillance, medical imaging and microscopy, however in image retrieval most methods are focussed on higher resolution images. Most image retrieval methods use SIFT (like) features [32], which do not appear frequently enough in low resolution images to be useful. Another problem in image retrieval is that most bag-of-feature methods [28] [25] perform quantization using a training set. Using vector quantization, a lot of information about the original features is lost and unseen features in the training set might not be represented at all. Of course, other features can be extracted from the low resolution images. This however often requires domain specific knowledge about which features are important. In image retrieval, this knowledge is not always available and a training dataset to obtain this knowledge is usually not representative enough. The purpose of image retrieval is often to discover if similar images of unseen categories are in large datasets, where there is no knowledge of these unseen categories anyway.
In order to quickly search these large datasets, several approximate nearest neighbor search

algorithms can be applied to solve this problem. Examples of approximate nearest neighbor search methods are the kd-tree used in [32] and the inverted file system [25]. These approaches can achieve good results if there are distinctive SIFT features, which is not the case for low resolution images. Locality Sensitive Hashing (LSH) [16] [2] allows us to retrieve feature vectors with a small Euclidean distance from the query feature vector. Because there are not many distinctive features, LSH allows us to search for combinations of features, which are probably distinctive. However, the underlying Euclidean distance between feature vectors used in LSH does not allow us to indicate the importance of certain features.
There are many improvements for LSH. Semantic hashing is proposed in [42], where each item in that dataset is represented by a binary code and a feedforward neural network is trained to calculate the binary code for a novel input. Other machine learning approaches are pursued to obtain compact codes by using Adaboost [31]. Recently, Spectral Hashing [54] was proposed to obtain optimal bitstrings given a training set of feature vectors. One problem with image retrieval is that it is very difficult to obtain a training set that is representative of the entire dataset.
For this reason, our method uses a different distance measure that does not rely on training. This work is inspired by the Information Bottleneck described in [31] and applied to image clustering in [17], where sets of features are compared instead of feature vectors. There is other related work using the Information Bottleneck for feature selection to cluster videos in [23] and to improve quantization of the codebook for image retrieval in [27]. Our method uses the approach presented in [17], and our main contribution is to extend this so it can be used in image retrieval, explaining how to obtain binary codes that allow us to index the images. We show that the information bottleneck can be used with low resolution features (like colour, texture and contours) to perform image retrieval on low resolution images.

3.3 Methodology

The Information Bottleneck is the underlying theory behind this method, where it is used as a distance measure between images. To use this distance measure, we need to obtain sets of features from the images and models that can describe the features. Gaussian Mixture Models obtained from the feature sets allow us to compute the distance measure between images, which can be used to rank the images. However, in image retrieval, datasets are often too large to compute the distance between all images with a query image. For this reason, hash functions (LSH) are proposed to compute codewords, which allows fast indexing of the dataset. Given a query, fast retrieval of a subset of promising images is then possible, so only the images in the selected subset are ranked using the Information Bottleneck distance measure.

3.3.1 Information Bottleneck

In this work, the Information Bottleneck principle is used as a distance measure between images. We were inspired by [17] and use a similar notation in order to explain this algorithm. Given a joint distribution $p(x, y)$ on the "model" space X and the "feature" space Y, Goldberger et al. [17] find a clustering $\hat{X}$ that minimizes the information loss $I(X;\ Y) - I(\hat{X};\ Y)$, where $I(X;\ Y)$ is the mutual information between X and Y. In this work, the information loss is used as a distance measure to obtain the ranking between a queried image and retrieved images.

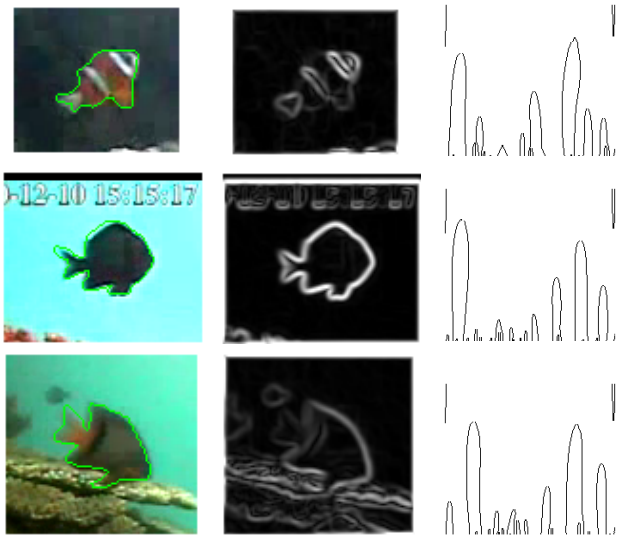


Figure 12: The color, texture and contour features of the fish images: In the columns, the segmented fish images, the magnitude of the Canny edge detector and the Curvature Scale Space of the contour are shown respectively.

Given two models x_1 and x_2, the information loss due to merging the models is given by:

$$d(x_1, x_2) = I(X_{\textbf{before}};\ Y) - I(X_{\textbf{after}};\ Y) \tag{11}$$

In this case, $I(X_{\textbf{before}};\ Y)$ gives the mutual information of each model describing the feature before merging the models. $X_{\textbf{before}}$ can be seen as modeling the features with $p(y|x_1)$ and $p(y\ |x_2)$ separately and $I(X_{\textbf{after}};\ Y)$ is the mutual information afterwards. $X_{\textbf{after}}$ can be seen as modeling the features if you combine $p(y|x_1 \cup x_2)$. According to Goldberger et al. [17] this gives:

$$\begin{aligned} d(x_1, x_2) &= \sum_{y,i=1,2} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} \\ &\quad - \sum_{y} p(x_1 \cup x_2, y) \log \frac{p(x_1 \cup x_2, y)}{p(x_1 \cup x_2)p(y)} \qquad (12) \\ &= \sum_{i=1,2} p(x_i) D(p(y|x_i) \| p(y|x_1 \cup x_2)) \qquad (13) \end{aligned}$$

In this case, $D(f\|g)$ is the Kullback-Leibler (KL) divergence and this equation is similar to the Jenson-Shannon divergence and can be used as a distance measure. In the next section, we will discuss how to obtain both the features and the models from the fish images.

3.3.2 Feature Extraction

In order to compare fish images, there are three important features according to biologists, namely the color of the fish, the texture of the fish (stripes, spots, etc) and the contour of the fish. We show that all these different features can be converted into the same representation. We start with a set of segmented fish images from a fish detection algorithm [47], shown in

Figure 12. We compute the coordinates of the bicone HSL (Hue,Saturation,Light) colour space for all segmented pixel values, which gives a set of color values together with their image locations. The location values of the colors are normalized based on the center and size of the segmentation. For the texture, we compute the Canny edge detector on all pixel values, giving us a set of magnitude and orientation values of the edges together with their normalized location values. Given the contour of the fish, we compute the Curvature Scale Space (CSS) described in [1] giving us the graph shown in Figure 12 (third column). At the moment, the fish images are not rotated or flipped in order to obtain a standard swimming direction, because we assume that the dataset is large enough to find the same fish in almost any similar swimming direction. These feature sets are modeled by a mixture of Gaussians like [17]. A set of features $Y^{\pi} = \{y_1, ..., y_n\}$ is coupled for each different feature space π i.e. colour, texture, contour. For the different feature spaces, both the dimensions of y_i and the number of values $\|Y\|$ can be different. For clarity, we give an intuition of our method on a single feature set and for this reason leave out π in most equations. We model each set of features with a Gaussian Mixture Model (GMM) $f(y|x)$. Using the Expectation-Maximization described in [56], the GMMs are computed using the Minimum Description Length to determine the number of Gaussians k.

$$f(y|x) = \sum_{j}^{k_x} \alpha_{x,j} N(y, \mu_{x,j}, \Sigma_{x,j}) \tag{14}$$

Equation 14 gives the Gaussian Mixture Model, k_x is the number of mixtures for model x. The variables $\alpha_{x,j}$, $\mu_{x,j}$, $\Sigma_{x,j}$ are respectively the weight, the mean and covariance for each Gaussian, which are estimated from the values Y.

Figure 13 shows the GMMs obtained from the color values of different fish images. These are however five dimensional features (color and location values). The second column in Figure 13 shows the Gaussians as ovals, where the color is the mean color of that Gaussian at its normalized position. In the third column in Figure 13, the three dimensional bicone HSL space is shown where we plot an ellipsoid for each Gaussian with its mean color. For texture, we can perform the same operation as for colour because it is on a pixel basis, however, converting the CSS into a GMM is not obvious. For the CSS, the region under the curves are filled (Figure 14 shows that the CSS curves look similar to a Mixture of Gaussians). By sampling from these curves, we obtain a GMM which models the CSS. Using the GMM, both features Y and models X are obtained which makes it possible to use the information loss described in the previous section.

To compute the similarity between fish, we can now compute the similarity between their models x_1, x_2 on the data y. This distance can be written as following:

$$d(x_1, x_2) = \sum_{i=1,2} D(f(y|x_i) \| f(y|x_1 \cup x_2)) \tag{15}$$

In this case, we assume that a uniform prior probability on $p(x_i)$ (see Equation 13) allowing us to leave this term out of Equation 15 . This distance measure can also be used for more than one image by combining GMMs. This gives the flexibility to compare sets of images with each other using this distance measure. In the case of video surveillance, objects are usually visible in multiple frames which allows us to combine the appearances of the objects in multiple frames.

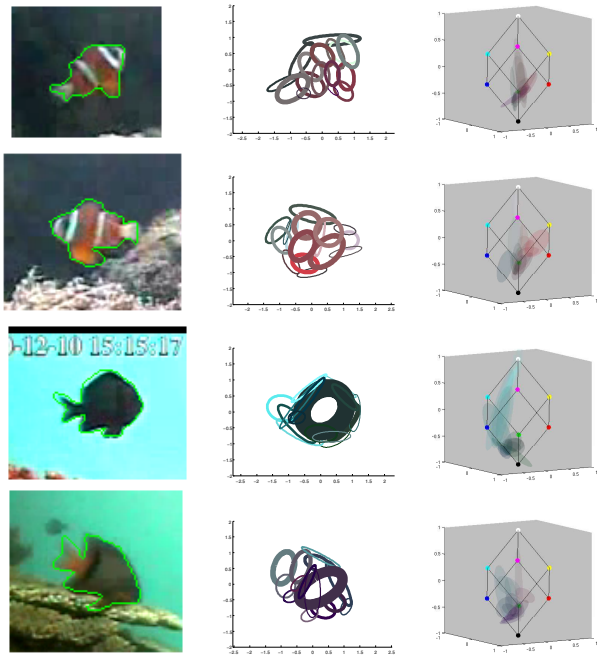


Figure 13: The Gaussian Mixture Models of the colors. The fish images are in the first column. The second column shows the GMMs where the oval indicate the mean and covariance in position combined with average color of each Gaussian and where the thickness of the lines indicates the weight $\alpha_{x,j}$ of the individual Gaussians. The third column shows the ellipsoid for each Gaussian in the bicone shaped HSL color space, where the cube corners indicates the locations of some primary colors.

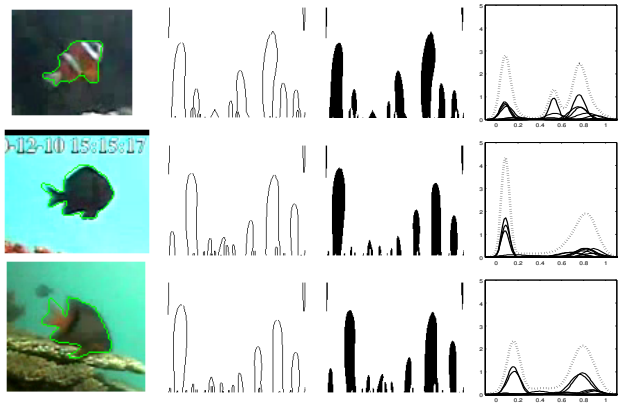


Figure 14: The Gaussian Mixture Models of the Curvature Scale Space of the fish image (first column). In the second and third column, the original CSS representation and the filled curves are shown. The final column gives the estimated GMMs based on the filled curves.

To compute the KL divergence between two mixtures of Gaussians f and g, a Monte-Carlo simulation is used, because there is no closed-form expression. The KL-divergence for f and g is given by

$$D(fg) = \frac{1}{n}\sum_{t=1}^{n}\log\frac{f(y_t)}{g(y_t)} \tag{16}$$

where $y_1, ..., y_n$ are sampled from the GMM $f(y)$. Notice that this gives the ability to compute a distance between two feature sets modeled by GMMs. To incorporate multiple feature spaces, the sum is taken over all the distances between the different sets of features and models. However, computing the distance between each image in the dataset is not feasible in a large dataset, so we use Locality Sensitive Hashing for indexing.

3.3.3 Codeword extraction

In Locality Sensitive Hashing (LSH), there are multiple ways to compute a codeword from a feature vector. One of the most common ways to obtain a single bit given the feature vector $\mathbf{v}$ is $h(\mathbf{v}) = \lfloor\frac{\mathbf{a}\mathbf{v}+b}{R}\rfloor$. In this case, both the vector $\mathbf{a}$ and b are randomly chosen for each bit, where $\mathbf{a}$ is vector with the same dimension as $\mathbf{v}$ from a stable distribution and b is a real number from a uniform distribution over the range $[0, R]$ and R is the expected range of the feature vector. In [54], the following requirements for the function h to compute the bits are given: (1) the function has to be easily computed for new images, (2) the function requires small numbers of bits to code the full dataset and (3) similar items get similar codewords by the function.
In the previous section, a GMM x is computed for every fish image. Given a random point y_r in the feature space Y, we can compute the probability $f(y_r|x) > t$ and together with a threshold t, this will give us a binary decision, performing this operation multiple times gives us codewords. This already satisfies both requirements (1) and (3). Because the computation of the probability given a GMM is very easy and, given similar fish images, we expect that if the GMMs of these fish images are quite similar that this also gives a similar probability. A more formal definition of our hash function is given as follows:

$$h(f) = \log f(y_r^{\pi_r}|x^{\pi_r}) > t \tag{17}$$

For the hash function $h(f)$, r denotes index of a randomly properties that are randomly chosen, meaning $y_r^{\pi_r}$ is a single random feature value of a certain feature set of a randomly chosen feature space π_r (i.e. colour, texture, contour) and x^{π_r} is the GMM that models that randomly chosen feature space. In LSH with a feature vector $\mathbf{v}$, a hash function $h(\mathbf{v})$ is used to index images allowing the fast retrieval of other feature vectors with a small Euclidean distance. In this method, the distance between GMMs f is used, so a hash function $h(f)$ is computed for indexing which allows fast retrieval of potentially similar GMMs. The log of the GMM is used in the hashing function and the threshold t can be set randomly. However, better results can be obtained by using a set of images to obtain a threshold t where $h(f)$ has 50 % chance of being zero or one, making the bit more efficient which allows us to satisfy requirement (2).
One of the advantages of this method is that there is a clear relationship between the bit and the feature space, because the bit encodes the likelihood that a certain random feature y_r is part of the model x. When using LSH [2], instead of extracting a single bit, multiple bitstrings are extracted. Similar bitstrings mean that multiple features are represented similarly by the models. Based on L bitstrings each with length of K, the nearest neighbor search with LSH

is performed. Here we give a brief overview of LSH [2]: For each bitstring, a hash table is created allowing fast retrieval of the same bitstring. However because images are never exactly the same, bitstrings might be slightly different, so searching L bitstrings enlarges the chance of finding similar images, where both K and L are parameters that can be tuned. Given the set of retrieved images using this algorithm, which is a small subset of the entire set, we compute information loss given in Equation 15 between all those retrieved images and the query image allowing us to rank the images based on the information loss.

3.4 Experiments

3.4.1 Comparison

The Information Bottleneck method is compared to two other methods: the first method is the standard bag-of-features method using the VLFeat implementation [53] to obtain a normalized histogram of SIFT features. This normalized histogram is then used as a feature vector into LSH and the Euclidean distance is computed to determine the ranking. The problem is however that in low resolution images (especially the fish images used in this work) there are often only a small number of SIFT features and in some cases there are no features at all. We decided not to query the images where no features are detected, while for the other methods, all the images in the dataset are queried. The second method is LSH combined with a feature vector made up of properties used for fish recognition. In this case, the shape of the fish is used to determine the head and tail and align the fish images. Using both more specialised texture and contour features of the aligned fish and color histograms in certain regions (head, tail, top, bottom, entire segmentation), a feature vector is created to describe each fish. This feature vector is then used in LSH and the Euclidean distance is used to rank the fish images. For LSH, the bitstring length and number of hash tables are respectively, $K = 24$ and $L = 30$, and are used for both the described methods above and the Information Bottleneck method.
We created two versions of the Information Bottleneck method for image retrieval. The first version of this method uses only a single GMM of colour (HSL) values together with their normalized location values in the images. The second version of this method uses multiple GMMs, which include both the colour values with their normalized location values, texture (Canny) with normalized location values and contours (CSS).

3.4.2 Dataset

The dataset is a low resolution fish image database, where the average fish image size is 78×82 and the average numbers of pixels of the segmented fish is 1003. These images are obtained from underwater cameras which are monitoring a coral reef environment. These cameras are recording every day, giving us a very large database of interesting video material. Background subtraction together with tracking techniques are used to extract fish images from the videos [47]. 1955 trajectories of fish are detected. This database has 20074 labeled fish images, where a lot of images (11310) are also annotated as "bad image", because of blurring effects in the water and the low resolution of fish in the images which made it impossible to determine the exact species. There are 43 different fish species in this database, where the distribution of species varies greatly. A couple of species appears very often (e.g. clownfish), but there are also around 20 very rare species. The dataset is in reality much larger than the

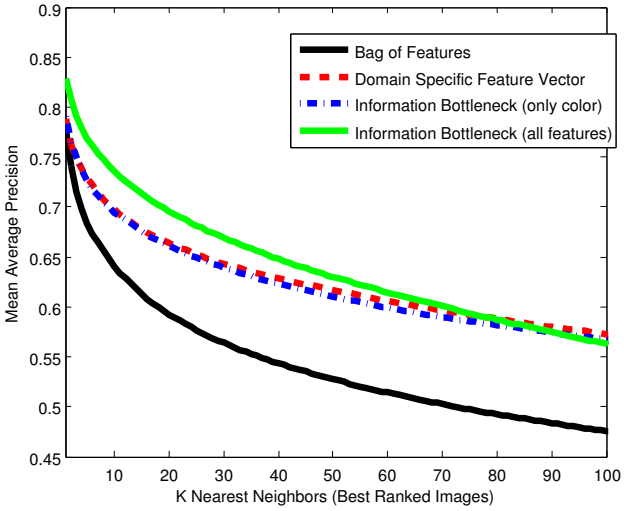


Figure 15: The mean average precision of all methods, where the Information Bottleneck with all features performs best.

20074 fish images, but this number of images is annotated at the moment.

3.4.3 Methodology of Experiments

In the previous section, an information retrieval method is described, where we are interested in the number of similar fish which can be found by querying with a certain fish image. In our method, the fish images are ranked, where we like to obtain images of the same species in the high ranks if we search for a certain species. There are also many "bad images" in the database, so if we search for a bad image, we would like to retrieve "bad images" instead of fish allowing us to discard these images. In order to evaluate these image retrieval algorithms, we query with all the images of a trajectory in the entire dataset, where we perform a leave-one-out experiment removing all images of that trajectory from the dataset. Querying with all images of a trajectory makes all methods more robust against the very uncontrolled movements of the fish. To compute the distances of the multiple fish images in the trajectory, we sum over the individual distances for all methods. The mean average precision curves are calculated for all methods and are plotted in Figure 15, which shows the average precision in retrieval given the best k ranked images for different values of k. The ideal retrieval function has precision 1.0 for all values of k.

3.4.4 Results

Figure 15 shows that the Information Bottleneck on all features performs better than the other compared methods. In the case of the Information Bottleneck on all features, subsets from the LSH algorithm (see Section 3.3.3) are obtained with an average of 148 fish images in them. The feature vector created for fish recognition is stable and works well with the Euclidean distance. This feature vector, however, takes into account a lot of domain specific assumptions. The Information Bottleneck on the colour set has almost the same performance by only taking into account the pixel values and positions in the image, which did not require any domain specific

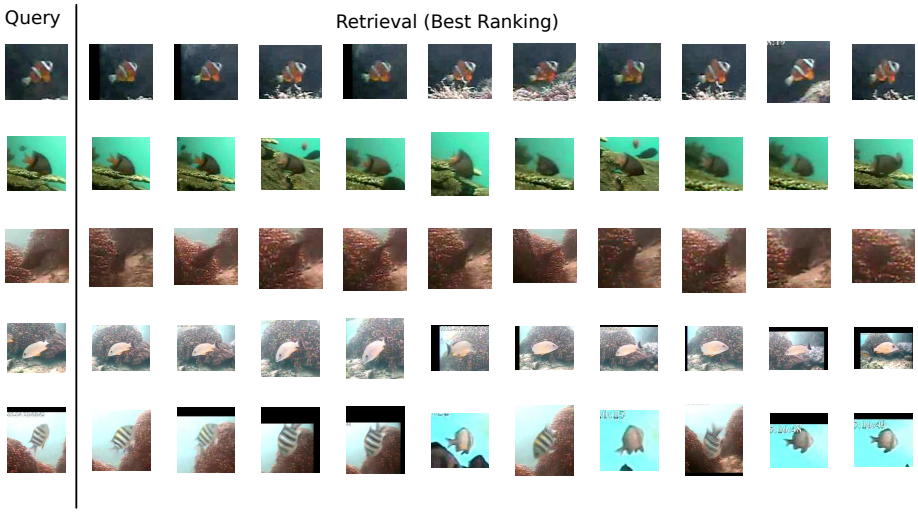
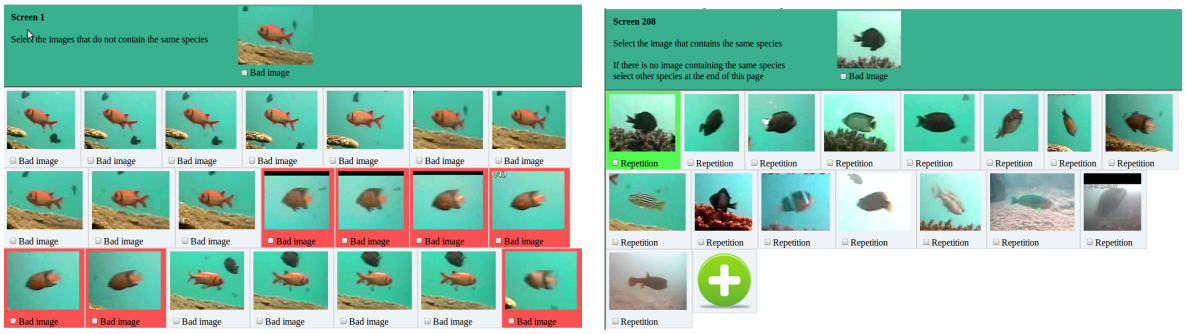


Figure 16: Some results given the query images on the left hand side: The first rows shows the 10 neighbors of the clown fish image (there are many images of clown fish in our database), the third row show results of querying a "bad image", the final row shows that incorrect fish images are also sometimes found.

assumptions. The performance of SIFT with the bag-of-features method indicates that another direction was necessary to solve the problem of image retrieval on low resolution fish images. In Figure 16, the results of the queries (10 best ranked images) for a single fish image are shown, where there are both query results on fish images (first two and last two rows) and a "bad image" (middle). The annotation of "bad image" is important, because not all images are useful for species classification and this allows us to estimate which images are good enough. The clown fish appears very often in our dataset, which is why the retrieved images look very similar to the query image, although they are from different trajectories. The similarity arises because: 1) many fish, 2) fixed camera, 3) typical fish behaviour, 4) location based feature model. The two fish at the bottom of Figure 16 are less common. The query time of the method depends on the size of the dataset, however querying from the 20074 fish image is on the order of milliseconds. Queries on a dataset of 393.101 fish images take around a second using a single desktop computer.

3.5 Application

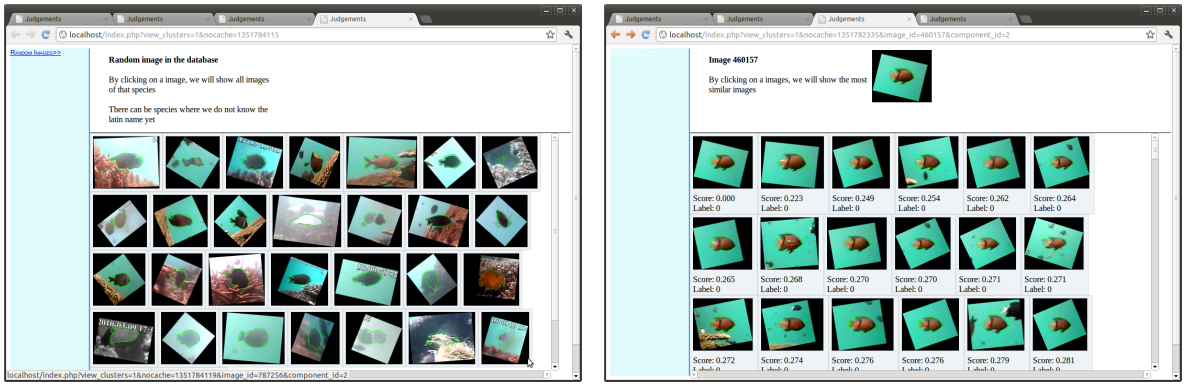
In the fish4knowledge project, fish clustering is mainly used for annotation of fish species. A web interface has been developed to support the annotation of different species. Some screenshots of the two main interfaces are shown in Figure 17(a) and 17(b). For the first interface, we use the cleaning interface shown in Figure 17(a). In this case, the representative cluster image is shown at the top of the screen and the rest of the images in that cluster are put under this image. The user only has to select the images which are not correctly clustered (where the fish species label is not similar to the label in the representative cluster image) and continue to the next interface. In the second interface, users link the clusters to labels using the representative images (image on top in first interface). Notice that by linking these images, we also immediately link the images that belong to the underlining clusters. The second interface shown in Figure 17(b) is used to link the representative image either to one of the previous



(a) The first interface to remove images from the cluster by clicking on the image that does not belong to the same label as the representative image in the top row

(b) The second interface to link the image in the top row to a label by clicking on one of the gallery images which belonging to the same label or add a new label by pressing the green plus button

Figure 17: Annotation interfaces



(a) The first interface allows you to choose a random fish to see if similar fish can be found in the dataset

(b) The second interface put the selected picture of the fish on top and shows in an ordered way similar fish that in found in the dataset. Average query time is around 1 second

Figure 18: Query interfaces

representative images of a label or the user will create a new label by pressing the green plus button.

At the moment, these clusters are determined offline and are loaded into a database. We have however developed a webinterface (Figure 18) for searching for similar fish by clicking on the fish images which are interesting. The first interface (Figure 18(a)) in this case shows a random selection of the fish in our dataset, by clicking on a fish image the method searches for all the similar fish in a dataset of 393101 image of fish (Figure 18(b)). The normal query time is around 1 second which is what is expected from these kind of webpages. In the future, the plan is to combine both interfaces, allowing us to search and label at the same time, so we can focus more on the uncommon fish.

4 Conclusion

In this document, we have presented both a fish recognition and clustering method. Both methods are already used in the Fish4Knowledge system, where we currently perform fish recognition in our system based on the methodology presented in Section 2, which has already processed 18,475,001 fish images (result from 1 November 2012), which are stored in our database server in Taiwan. The fish clustering methods are actively being used for annotation, which provides us with a growing set of fish images for both training and evaluation purposes. At the moment, using clustering 303,625 annotations has been performed by users, however to obtain accurate annotation multiple annotation of users have to be combined giving us a set of around 28,264 usable annotated fish images covering about 40 species. In this case, we ignore the images that are filtered out for reasons of occlusions by coral or other fish, bad quality images, etc.

4.1 Future Work

In our future work, we hope to improve the accuracy of both methods even further where especially the increasing size of annotated images help us to make more robust classifiers for fish recognition. Another topic which requires more work is the discovery of species that have not been seen before by our classification methods. In the recorded videos, certain fish are very common, while a large percentage of the fish is so rare that we only have a couple of recordings. Techniques are being developed to find the rare fish (so to not classify them incorrectly) and using nearest neighbor search to find similar fish images allowing us to build models for these kinds of fish to classify them in the future.

References

[1] Sadegh Abbasi, Farzin Mokhtarian, and Josef Kittler. Curvature scale space image in shape similarity retrieval. *Multimedia Syst.*, 7(6):467–476, November 1999.

[2] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008.

[3] D. R Blidberg. The development of autonomous underwater vehicles (AUVs); a brief summary. In *IEEE International Conference on Robotics and Automation*, volume 4, 2001.

[4] B. Boom and R. Fisher. Fish4Knowledge deliverable d5.1 component interface and integration plan, 2011.

[5] Bastiaan J. Boom, Phoenix X. Huang, Jiyin He, and Robert B. Fisher. Supporting ground-truth annotation of image datasets using clustering. ICPR 2012, to appear.

[6] P. Brehmer, T. D. Chi, and D. Mouillot. Amphidromous fish school migration revealed by combining fixed sonar monitoring (horizontal beaming) with fishing data. *Journal of Experimental Marine Biology and Ecology*, 334(1):139–150, June 2006.

[7] M. J. Caley, M. H. Carr, M. A. Hixon, T. P. Hughes, G. P. Jones, and B. A. Menge. Recruitment and the local dynamics of open marine populations. *Annual Review of Ecology and Systematics*, 27:477–500, January 1996.

[8] Silla Carlos and Freitas Alex. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2010.

[9] Chang Chih-Chung and Lin Chih-Jen. LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):1–27:, 2011.

[10] G. C. Daily. *Nature's services: societal dependence on natural ecosystems*. Island Pr, 1997.

[11] Jia Deng, Alexander Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, volume 6315, pages 71–84. 2010.

[12] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: a large-scale hierarchical image database. CVPR, pages 248 –255, 2009.

[13] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *IEEE International Conference on Robotics and Automation*, page 2091–2098, 2006.

[14] Y. Freund and R. E. Schapire. A Decision-Theoretic generalization of on-Line learning and an application to boosting. *Computational Learning Theory*, 904:23–37, 1995.

[15] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972 –976, February 2007.

[16] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[17] J. Goldberger, S. Gordon, and H. Greenspan. Unsupervised image-set clustering using an information theoretic framework. *IEEE Trans. on Image Processing*, 15(2):449–458, February 2006.

[18] R. M. F. Gomes, J. B Sousa, and F. L Pereira. Modeling and control of the IES project ROV. In *European Control Conference*, pages 3436–3441, Cambridge, UK, 2003.

[19] A. D Gordon. A review of hierarchical classification. *J. Royal Stat. Soc.*, 150(2):119–137, 1987.

[20] G. G. Hall. *Remote environmental sensor array system*. PhD thesis, Queen's University, 2007.

[21] S. Harsdorf. Contrast-enhanced optical imaging of submersible targets. In *Proceedings of SPIE*, pages 378–382, Munich, Germany, 1999.

[22] X. C. He and N. H. C. Yung. Curvature scale space corner detector with adaptive threshold and dynamic region of support. In *Pattern Recognition, International Conference on*, volume 2, pages 791–794, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[23] Winston H. Hsu and Shih-Fu Chang. Visual cue cluster construction via information bottleneck principle and kernel density estimation. In *Proceedings of the 4th international conference on Image and Video Retrieval*, CIVR'05, pages 82–91, Berlin, Heidelberg, 2005. Springer-Verlag.

[24] K. Ishii, H. Takahashi, K. Oda, T. Kojima, H. Soeda, K. Takemoto, M. Hiwada, and S. Sameshima. 3-D analysis of behavioral responses of aquatic life using stereo video imagery. In *OCEANS'98 Conference Proceedings*, volume 1, page 267–271, 1998.

[25] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *Int. J. Comput. Vision*, 87(3):316–336, May 2010.

[26] R. Larsen, H. Ólafsdóttir, and B. Ersbøll. Shape and texture based classification of fish species. In *Proceedings of the Scandinavian Conference on Image Analysis*, page 745–749, 2009.

[27] Svetlana Lazebnik and Maxim Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(7):1294–1309, July 2009.

[28] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[29] D. J. Lee. Contour matching for a fish recognition and migration-monitoring system. In *Proceedings of SPIE*, volume 5606, pages 37–48, Philadelphia, PA, USA, 2004.

[30] R. Li, H. Li, W. Zou, R. G Smith, and T. A Curran. Quantitative photogrammetric analysis of digital underwater video imagery. *Oceanic Engineering, IEEE Journal of*, 22(2):364–375, 1997.

[31] Yingyu Liang, Jianmin Li, and Bo Zhang. Learning vocabulary-based hashing with adaboost. In *Proceedings of the 16th international conference on Advances in Multimedia Modeling*, MMM'10, pages 545–555, Berlin, Heidelberg, 2010. Springer-Verlag.

[32] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[33] Charles Mathis. Classification using a hierarchical bayesian approach. volume 4 of *Proc. of ICPR*, pages 103–106, 2002.

[34] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1376–1381, 1998.

[35] G. Nadarajan, Y. H Chen-Burger, and R. B Fisher. A knowledge-based planner for processing unconstrained underwater videos. In *IJCAI'09 Workshop on Learning Structural Knowledge From Observations (STRUCK'09)*, 2009.

[36] G. Nadarajan, Y.-H. Chen-Burger, R.B. Fisher, and C. Spampinato. A flexible system for automated composition of intelligent video analysis. Proc. of ISPA, pages 259–264, 2011.

[37] M. Okamoto, S. Morita, and T. Sato. Fundamental study to estimate fish biomass around coral reef using 3-dimensional underwater video system. In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 2, page 1389–1392, 2000.

[38] J. P. Queiroz-Neto, R. Carceroni, W. Barros, and M. Campos. Underwater stereo. In *Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium*, page 170–177, Washington, DC, USA, 2004. IEEE Computer Society. ACM ID: 1026250.

[39] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics (TOG)*, pages 309–314, 2004.

[40] A. Rova, G. Mori, and L. M. Dill. One fish, two fish, butterfish, trumpeter: Recognizing fish in underwater video. In *IAPR Conference on Machine Vision Applications*, pages 404–407, 2007.

[41] B. P. Ruff, J. A. Marchant, and A. R. Frost. Fish sizing and monitoring using a stereo image analysis system applied to fish farming. *Aquacultural engineering*, 14(2):155–173, 1995.

[42] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009.

[43] R. A. Salam, A. O. Ee, and M. S. Hitam. *Unsupervised Color Correction Using Cast Removal for Underwater Images*. World Scientific and Engineering Academy (WSEAS) Transactions on Information Science and Applications, 2004.

[44] Y. Y. Schechner and N. Karpel. Clear underwater vision. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 536–543, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[45] R. Schettini and S. Corchs. Underwater image processing: state of the art of restoration and image enhancement methods. *EURASIP J. Adv. Signal Process*, 2010:14:1–14:7, January 2010.

[46] 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'08)*, page 514–519, 2008.

[47] C. Spampinato, D. Giordano, R. Di Salvo, Y. H.J Chen-Burger, R. B. Fisher, and G. Nadarajan. Automatic fish classification for underwater species behavior understanding. In *Proceedings of the first ACM international workshop on analysis and*

retrieval of tracked events and motion in imagery streams, page 45–50, New York, NY, USA, 2010.

[48] N. J. C. Strachan. Length measurement of fish by computer vision. *Computers and electronics in agriculture*, 8(2):93–104, 1993.

[49] N. J. C Strachan. Recognition of fish species by colour and shape. *Image and Vision Computing*, 11:2–10, January 1993. ACM ID: 156583.

[50] N. J. C. Strachan, P. Nesvadba, and A. R. Allen. Fish species recognition by shape analysis of images. *Pattern Recognition*, 23(5):539–544, 1990.

[51] Y. H. Toh, T. M. Ng, and B. K. Liew. Automated fish counting using image processing. In *International Conference on Computational Intelligence and Software Engineering*, page 1–5, 2009.

[52] L. A. Torres-Méndez and G. Dudek. A statistical learning-based method for color correction of underwater images. *Research on Computer Science*, 17, 2005.

[53] Andrea Vedaldi and Brian Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM.

[54] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *Neural Information Processing Systems Conference*, pages 1753–1760, 2008.

[55] B. Zion, A. Shklyar, and I. Karplus. In-vivo fish sorting by computer vision. *Aquacultural Engineering*, 22(3):165–179, June 2000.

[56] Z. Zivkovic and F. van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651–656, May 2004.

