Approximate Nearest Neighbor Search to support Manual Image Annotation of large domain-specific datasets

Bastiaan J. Boom Phoenix X. Huang Robert B. Fisher
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
bboom@inf.ed.ac.uk

ABSTRACT

The annotation of large datasets containing domain-specific images is both time-consuming and difficult. However, currently computer vision and machine learning methods have to deal with ever increasing amounts of data, where annotation of this data is essential. The annotated images allow these kind of methods to learn the variation in large datasets and evaluate methods based on large datasets. This paper presents a method for annotation of domain-specific (fish species) images using approximate nearest neighbor search to retrieve similar fish species in a large set (216,501) of images. The approximate nearest neighbor search allows us to find a ranked set of images in large datasets. Presenting similar images to users allows them to annotate images much more efficiently. In this case, our user interface present these images in such a way that the user does not need to have knowledge of a specific domain to contribute in the annotation of images.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*Clustering, Information filtering*; H.3.3 [**Information Search and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; H.2.8 [**Database Management**]: Database Applications—*image databases*

Keywords

manual image annotation, approximate nearest neighbor search, large-scale clustering of fish images

1. INTRODUCTION

Manual annotation of domain specific images is necessary allowing computer vision methods to learn the domain knowledge in the images and to evaluate the performance of these methods. To annotate these kinds of images often experts are needed who can determine the appropriate labels. This paper discusses a method to build a large manually labelled dataset of images (e.g. underwater images of fish) together with groundtruth classifications (e.g. fish species) in an efficient manner without using expert knowledge. In the recent work of [3], clustering supports the annotation of domain specific images. Common clustering methods are often not able to cluster a large-scale dataset (100,000 images) in small enough portions. The new contributions of this paper are: First, using approximate nearest neighbors to support annotation by finding ranked clusters of similar images in domain-specific (fish species) large-scale datasets. Second, an approximate nearest neighbor search method that allows us to retrieve similar fish images for a large-scale datasets (216,501 fish images). Thirdly, an improved interface that uses the ranking ability of the approximate nearest neighbor search to allow the domain specific annotation in a more efficient way.

1.1 Related Work

Previous work in the manual annotation of images includes, for instance, the ESP Game [17] and LabelMe [13]. Most of this work focusses on internet images where often multiple tags can be defined for a single image. These approaches are suitable in the case of random internet images, but users often fail to give specific tags especially if domain knowledge is required. In [20], the problem of annotation of an image database of birds is solve by annotating multiple visual properties (like the color of tail, wings, beak) in order to obtain a specific label for the bird. The annotation of multiple properties might however not be the most efficient way to assign certain images to a specific class. Several methods combine user annotation and machine learning to obtain the groundtruth label, for instance [12]. The methods that are closest to our methodology are [18] and [3]. In [18], users can search and annotate images at the same time. This approach is developed for internet images, where all labels are defined apriori to the annotation. In [3], clustering is used to annotate the images, working with a smaller dataset however it is possible to add new labels. In our methodology, because nearest neighbor search is used instead of clustering, we are able to deal with large-scale datasets of images. We also make use of the fact that images are ranked allowing us to annotate only the interesting images closely ranked to the search class example.

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 [11] and

the inverted file system [8] and Locality Sensitive Hashing (LSH) [4][2]. 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 [14], 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 [10]. Recently, Spectral Hashing [19] 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 describe in [10] and applied to image clustering in [5], 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 [6] and to improve quantization of the codebook for image retrieval in [9]. Our approximate nearest neighbor search extends an approach presented in [5], 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 [5] can be used to perform image retrieval on low resolution fish images with features (like color, texture and contours) .

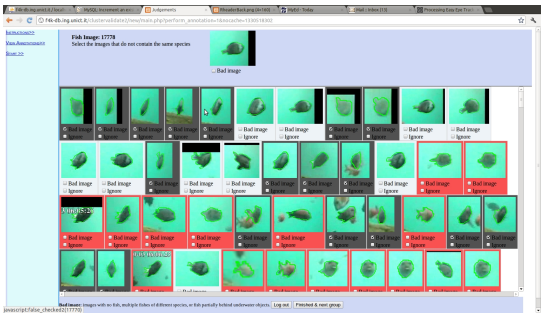
This paper is organised as follows: Section 2 explains the new user interface for the annotation of fish images. In Section 3, the nearest neighbor search methodology to obtained ranked clusters is described. Section 4 discusses the current implementation and Section 5 shows the results of the nearest neighbor search methodology and the number of annotations.

2. USER INTERFACES FOR ANNOTATION

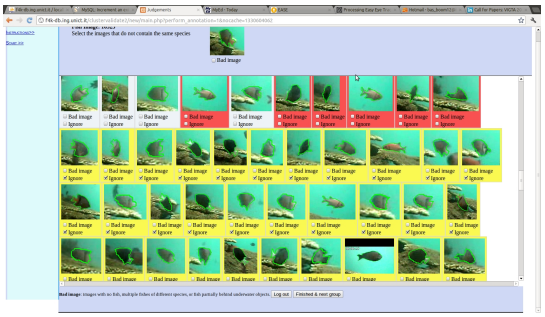
The annotation of thousands of images can be time consuming, where [3] already shows that the efficiency can be improved by clustering the images. In this paper, we improve the efficiency even more by using a ranked group of images. In our method, two interfaces are being used for the annotation of the images:

The first interface shown in Figure 1(a) allows users to remove images that do not belong to the same class as the representative image on top. In our case, there are two reasons that images have to be removed. The first reason is that the image belongs to another class while the second reason is that there is no fish inside the images because of a failure in the automatic fish detection methods. Clicking on the images indicate that they are of another class (using red surrounding around these images) and a checkbox "Bad Image" can be used in cases where the images did not contain a fish or the fish is not recognisable due to low resolution or image blur (using black surrounding around these images). The "Bad Images" are not shown again in the user interface.

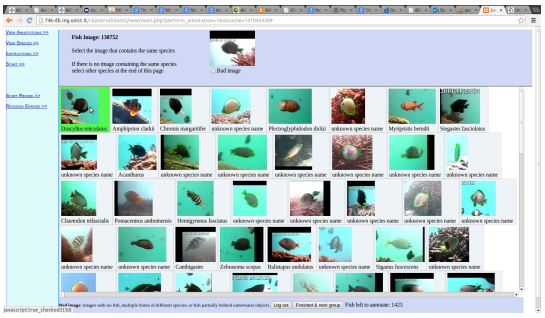
The second feature of the first interface is that the user can decide to stop annotation if lots of images are not similar to the fish image shown on top (Figure 1(b)). In our fish database, we have observed a lot of rare fish species that only appear for instance 20 times in our dataset, the



(a) **The first interface** to remove images from the cluster by clicking on the image which makes the surrounding of the image red. Also "bad images" can be removed by using a checkbox making the surrounding black.



(b) **The first interface** allows in the case of low ranked images to ignore them after a certain point making the rest of the images yellow.



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

Figure 1: Interfaces see Section 2 for more details.

nearest neighbor search will find more images, but in that case these images contain other species which are often of lower rank. By checking the ignore checkbox, all images with a lower ranking than the current image (less similarity to the representative image) are ignored for annotation and are not linked to the representative images. If under a certain ranking, more than 50% of the images are incorrect, it becomes more efficient to either annotate the correct images individually or to wait for a screen where these images are better represented, which is exactly what this feature allows users to do. The area surrounding these images will become yellow (see Figure 1(b))

The second interface (1(c)) is shown directly after completing the first interface (different from [3]), where the image on top stays the same. In the first interface, all images with a similar class are linked to the representative image on top. In the second interface, this group of images can be linked to the existing classes or a new class can be defined by linking to this representative images which are also shown in the first interface. In the case of linking the representative image to a class, we click on these representative images of this class making the surrounding area of the image green. A new label is created by pressing the green plus button. Currently, we ranked the classes in this interface by having the most common classes on top while having rarer classes lower. Experiments with performing ranking based on appearance of the classes were also performed, but users indicated that having the same ordering in the second interface is more user-friendly especially with a large amount of classes allowing them to remember already observed classes.

3. INFORMATION BOTTLENECK APPROXIMATE NEAREST NEIGHBOR SEARCH (IBANN)

The user interface uses an Approximate Nearest Neighbor Search method to obtain a cluster of ranked images given a representative image. The Information Bottleneck is the underlying theory behind our Approximate Nearest Neighbor Search method (IBANN), 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.

3.1 Information Bottleneck

In this work, the Information Bottleneck principle is used as a distance measure between images. We were inspired by [5] 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, 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

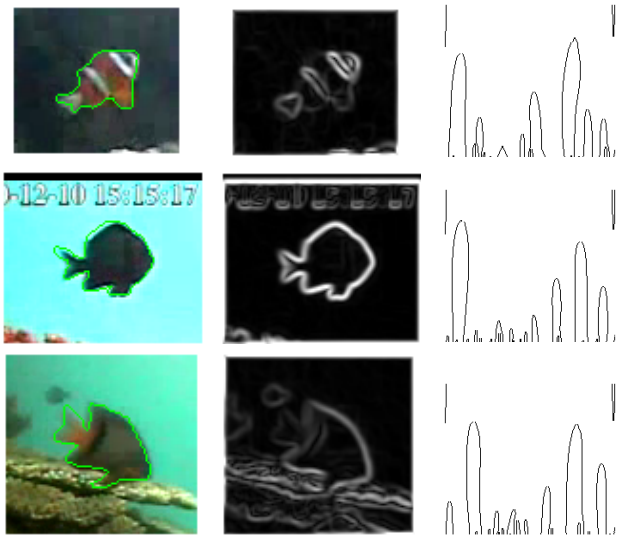


Figure 2: The color, texture and contour features of the fish images: In the columns, the segmented fish images, the magnitude of the Canny edge detector and the Curvature Scale Space of the contour are shown respectively.

to obtain the ranking between a queried image and retrieved images. Given two models x_1 and x_2, the information loss due to merging the models is given by:

$$d(x_1, x_2) = I(X_{\mathbf{before}}; Y) - I(X_{\mathbf{after}}; Y) \tag{1}$$

In this case, $I(X_{\mathbf{before}}; Y)$ gives the mutual information of each model describing the feature before merging the models ($X_{\mathbf{before}}$ can be seen as modeling the features with $p(y|x_1)$ and $p(y|x_2)$ separately) and $I(X_{\mathbf{after}}; Y)$ is the mutual information afterwards ($X_{\mathbf{after}}$ can be seen as modeling the features if you combine $p(y|x_1 \cup x_2)$). According to [5], this gives:

$$\begin{aligned} d(x_1, x_2) &= \sum_{y,i=1,2} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} \\ &\quad - \sum_{y} p(x_1 \cup x_2, y) \log \frac{p(x_1 \cup x_2, y)}{p(x_1 \cup x_2)p(y)} \quad (2) \\ &= \sum_{i=1,2} p(x_i) D(p(y|x_i) || p(y|x_1 \cup x_2)) \quad (3) \end{aligned}$$

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.

3.2 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. This paper shows that these different features can be converted into the same representation. We start with a set of segmented fish images from a fish detection algorithm [15], shown in Figure 2. We compute the coordinates

of the bicone HSL (Hue,Saturation,Light) color 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 2 (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 similar swimming direction. These feature sets are modeled by a mixture of Gaussians like [5]. A set of features $Y^{\pi} = \{y_1, ..., y_n\}$ is coupled for each different feature space π i.e. color, 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 [22], the GMMs are computed using the Minimum Description Length to determine the number of Gaussians k.

$$f(y|x) = \sum_{j}^{k_x} \alpha_{x,j} N(y, \mu_{x,j}, \Sigma_{x,j}) \tag{4}$$

Equation 4 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 3 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 3 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 3, 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 color 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 4 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. This distance can be written as following:

$$d(x_1, x_2) = \sum_{i=1,2} D(f(y|x_i)||f(y|x_1 \cup x_2)) \tag{5}$$

In this case, we assume uniform prior probability on $p(x_i)$ (see Equation 3) allowing us to leave this term out of Equation 5. 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.

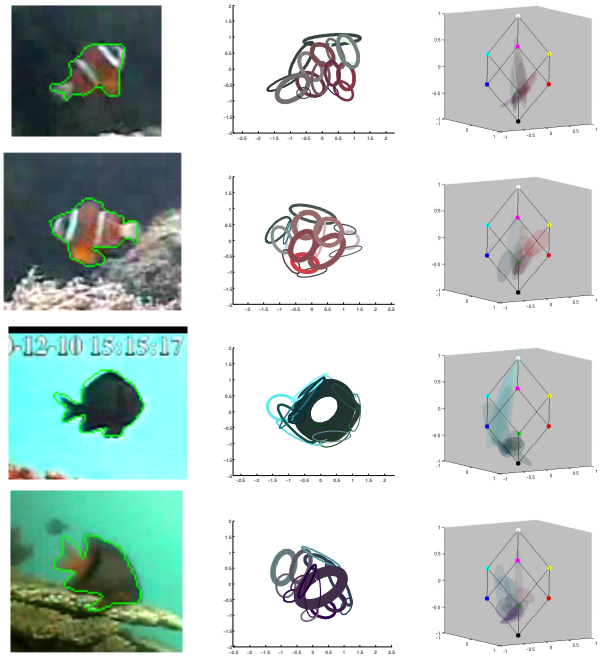


Figure 3: The Gaussian Mixture Models of the colors. The fish images are in the first column. The second column shows the GMMs where the oval indicate the mean and covariance in position combined with average color of each Gaussian and where the thickness of the lines indicates the weight $\alpha_{x,j}$ 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.

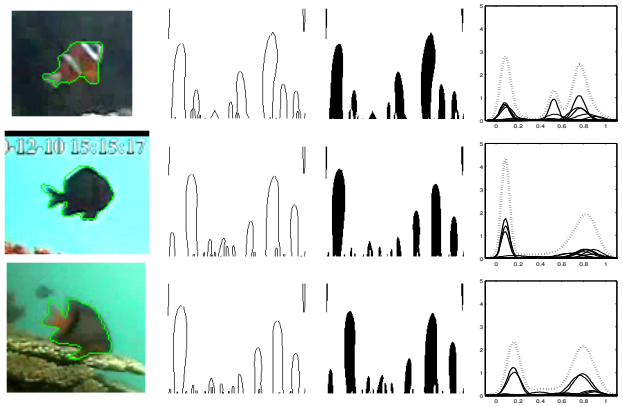


Figure 4: The Gaussian Mixture Models of the Curvature Scale Space of the fish image (first column). In the second and third column, the original CSS representation and the filled curves are shown. The final column gives the estimated GMMs based on the filled curves.

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

$$D(f||g) = \frac{1}{n}\sum_{t=1}^{n}\log\frac{f(y_t)}{g(y_t)} \quad (6)$$

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 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 a 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 [19], 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. 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:

$$h(f) = \log f(y_r^{\pi_r}|x^{\pi_r}) > t \quad (7)$$

For the hash function $h(f)$, r denotes 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. color, 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 and thus the obtained GMMs 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 5 between all those retrieved images and the query image allowing us to rank the images based on the information loss.

4. CURRENT IMPLEMENTATION OF ENTIRE SYSTEM

Currently, the approximate nearest neighbor search and the website for annotation are still separated, because the nearest neighbor search is developed in MATLAB which is difficult to connect to a webserver. To annotate the images, the nearest neighbors are precomputed for a large subset of representative images and stored in a MySQL database. The website is able to randomly select one of the representative images and get a sorted list of all the nearest neighbors. Currently, there are 4,553 representative images in the database for the 216,501 fish images that we have in total.

5. EXPERIMENTS AND RESULTS

For the experiments and results, we first focus on the results of the IBANN search method and compare them to some other methods. Afterwards, the number of annotations currently performed with this system are given, although more annotations are expected in the near future.

5.1 Approximate Nearest Neighbor Search

The IBANN search method is compared to two other methods: the first method is the standard bag-of-features method using the VLFeat implementation [16] 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 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 [7]. 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 IBANN search method.

We created two versions of the IBANN search method for image retrieval. The first version of this method uses only a

single GMM of color (HSL) values together with their normalized location values in the images. The second version of this method uses multiple GMMs, which include both the color values with their normalized location values, texture (Canny) with normalized location values and contours (CSS).

5.2 Dataset

The fish image database has an average fish image resolution of 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 [15]. Currently, 1955 trajectories of fish are used to verify performance of the Approximate Nearest Neighbor Search methods, containing 20074 labeled fish images. Many of these fish images (11310) are annotated as "Bad Image", because no fish is present or blurring effects in the water make 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 appear very often (e.g. clownfish), but there are also around 20 very rare species. The entire dataset is in reality much larger than the 20074 fish images, but this number of images is annotated and verified by us again to guarantee the quality of the annotation.

5.3 Methodology of Experiments

In this paper, 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 highest 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. To evaluate these algorithms, we query these search methods with all the images of a given trajectory in the entire dataset, where we perform a leave-one-out experiment removing all images of that trajectory from the dataset. Notice that automatic fish detection methods are used to obtain fish images for videos, where the fish is followed in multiple frames giving us a trajectory. 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 5, 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. In Figure 6, 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 fishes at the bottom of Figure 6 are less common.

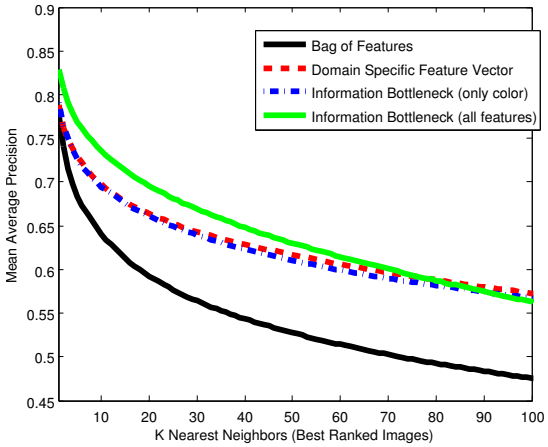


Figure 5: The mean average precision of all methods, where the IBANN search method with all features performs best.

5.4 User Interface and Obtained Data

Figure 5 shows that the IBANN search method on all features performs better than the other methods. It also shows that the precision for the first 100 neighbors is higher than 50%, which means that in the user interface less than 50% of the images have to be removed by the users. This has a direct effect on the efficiency of the users (already shown in [3] that good clustering results improve user efficient, allowing users to perform the same taske in a **third** of the time), although this is hard to measure exactly, because our users perform the annotation often in combination with other tasks. This makes a dedicated time measure of how long it takes to complete a screen impossible. However, our user interface presents ranked clusters, improving both the amount of images in the clusters and the quality of presentation of the clusters (making it probably even more efficient). Currently, we have 23 users who performed 408,871 annotations with a minimum of 66 annotations and maximum of 90,466 annotation per user. 91,894 images are annotated where a lot of images are labelled as "bad images", because no fish was present, low resolution, etc. For 28,264 images, we have obtained a species label, however these numbers are still increasing. An image with a species label is annotated by multiple users as the same fish species, where an agreement of 2 persons is necessary. More complex ways of determine user agreement can be used [21], however our main focus was on the creation of a large dataset of fish images for training and evaluation of our fish recognition methods.

6. CONCLUSION

A methodology is presented to annotate images more efficiently using Approximate Nearest Neighbor search. For a domain-specific dataset, we show that IBANN search methods gives good results (first 100 neighbors is higher than 55% precision) and performs better than other search methods on a subset of our dataset. Together with Approximate Nearest Neighbor search, a user interface is presented that makes use of both clusters and rankings obtained from the near-

Figure 6: Some results given the query images on the left hand side: The first rows shows the 10 neighbors of the clown fish image in order of similarity to the query image (there are many images of clown fish in our database). The third row show results of querying with a "bad image". The final row shows that incorrect fish images are also sometimes found (column 5,7,8 and 9).

est neighbor search. This method allows to efficiently and without much domain knowledge annotated the fish images by linking them to a fish species. This framework can also be used in other domains and with different nearest neighbor search methods. It allows users to create large datasets of groundtruth data for domain specific image recognition. These large datasets can be used to learn classifiers and verify performance of existing classifiers.

6.1 Future Work

The future plans are a full integration of both the Approximate Nearest Neighbor search and the website allowing for even more flexibility in labelling data, where in the case of observering an interesting fish species, the user can indicate that in the next screen he/she would like to have that image as the representative image. Also, integration of [21] on how to reach user agreements by making use of the user's performance compared to other users labelling similar images. Finally, we are investigating creation of a mechanism to filter out well-known classes while focussing on the rare classes in the dataset.

6.2 Acknowledgements:

This research was funded by the European Commission (FP7 grant 257024) and the Taiwan National Science Council (grant NSC100-2933-I-492-001) and undertaken in the Fish4Knowledge project (www.fish4knowledge.eu).

7. REFERENCES

[1] S. Abbasi, F. Mokhtarian, and J. Kittler. Curvature scale space image in shape similarity retrieval. *Multimedia Syst.*, 7(6):467–476, Nov. 1999.

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

[3] B. Boom, P. Huang, J. He, and R. Fisher. Supporting ground-truth annotation of image datasets using clustering. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1542–1545, 2012.

[4] A. Gionis, P. Indyk, and R. 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.

[5] 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, Feb. 2006.

[6] W. H. Hsu and S.-F. 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.

[7] P. X. Huang, B. J. Boom, and R. B. Fisher. Underwater live fish recognition using a balance-guaranteed optimized tree. In K. Lee, Y. Matsushita, J. Rehg, and Z. Hu, editors, *Computer Vision - ACCV 2012*, volume 7724 of *Lecture Notes in Computer Science*, pages 422–433. Springer Berlin Heidelberg, 2013.

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

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

[10] Y. Liang, J. Li, and B. 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.

[11] D. 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.

[12] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. Mach. Learn. Res.*, 99:1297–1322, 2010.

[13] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: A database and web-based tool for image annotation. *Int J of Computer Vision*, 77:157–173, 2008.

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

[15] 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*, ARTEMIS '10, pages 45–50, New York, NY, USA, 2010. ACM.

[16] A. Vedaldi and B. 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.

[17] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc of the SIGCHI conf on Human factors in computing systems*, CHI '04, pages 319–326, 2004.

[18] X.-J. Wang, L. Zhang, F. Jing, and W.-Y. Ma. Annosearch: Image auto-annotation by search. In *CVPR 2006*, volume 2, pages 1483 – 1490, 2006.

[19] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Neural Information Processing Systems Conference*, pages 1753–1760, 2008.

[20] P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *CVPR Workshops 2010*, pages 25 –32, june 2010.

[21] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.

[22] 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.