Vision based Analysis in Fishery Applications

Gaoang Wang

A dissertation

submitted in partial fulfillment of the

requirements for the degree of:

Doctor of Philosophy

University of Washington

2019

Reading Committee Members:

Dr. Jenq-Neng Hwang, Chair

Dr. Blake Hannaford

Dr. De Meng

Dr. Chun Yuan

Program Authorized to Offer Degree:

Department of Electrical and Computer Engineering

Abstract

Vision based Analysis in Fishery Applications

Vision-based analyses have drawn increasing attention since they enable a non-extractive and non-lethal approach to fisheries survey. The challenges of extracting information from vast amount of video or image data can be reduced by using automatic analysis techniques for segmentation, tracking, length/size measurement and species identification. The goal of the work is therefore to build a successful automatic development of these algorithms, which will greatly ease one of the most onerous steps in vision-based fisheries survey.

One of the major fishery applications is built for on-board chute-based images, related to fish species identification, segmentation and length/size measurement. Recently, image classification and recognition have been well studied in computer vision society with the rapid development of convolutional neural networks. However, there are still some challenges for species identification in fishery applications. For example, the fisheries surveys are conducted each year. Due to the environment change, the data acquired for each year may have large difference. As we know, for supervised learning method, if the testing data cannot be well represented by the training data, then the performance usually degrades on the testing set. On the other side, it is usually expensive to label all the data for each year. To address such issues, species identification with active learning approach is proposed for efficient labeling and model adaptation. We iteratively select informative samples for human labeling. The sample selection is formulated as a sparse modeling problem and efficient approximation solutions are proposed.

Another application on chute image dataset is fish size and length measurement which requires a reliable segmentation approach. However, there are two major challenges about the segmentation. One is that images may be blurred due to the spray of water on the camera lens, and the other is that some part of the fish body is out of the camera view. We present an innovative and effective contour-based segmentation refinement method to address such problems.

The other part of our completed system is fish abundance estimation based on the underwater videos from a remotely-operated vehicle (ROV). Once the video is captured, the live fish can be efficiently tracked and counted with our proposed system. The tracking and counting algorithm is followed by tracking-by-detection framework. The live fish are first detected by offline trained fish detector and TrackletNet tracker (TNT) is further developed for counting purpose. However, due to the diversity of fish poses, the deformation of fish body shape and the color similarity between fish and background, the detection performance greatly degrades, resulting in large error in tracking and counting. To deal with such issue, tracklet classifier is built for removing false positives, while TNT is trained to compensate the missing detections and ID switches. Finally, the tracking is conducted by a graph clustering method. This proposed strategy effectively addresses the false detection, occlusion, ID switches and largely decreases the tracking error.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to first express my sincere gratitude to my adviser, Prof. Jenq-Neng Hwang, for his guidance throughout my PhD study. He is always supportive and encouraging to help me pursue the goals and overcome the difficulties in research. From him I have learned a lot about how a successful scholar tackles open problems and gives good presentations.

I am grateful to my committee members, Prof. Chun Yuan, Prof. Blake Hannaford and Prof. De Meng, for the valuable suggestions they have given.

I am thankful to the alumni, visiting scholars and current colleagues I have met at the Information Processing Lab (IPL), Chun-Te Chu, Shian-Ru Ke, Xiang Chen, Kuan-Hui Lee, Younggun Lee, Jounsup Park, Zheng Tang, Renshu Gu, Tsung-Wei Huang, Li Chen, Haotian Zhang, Jiarui Cai, Yizhou Wang, Hao Xiao, Chengqian Ma, Xinyu Yuan, Hung-Min Hsu, Yanting Zhang, Fangyi Zhu, Ping Zhang, Aotian Zheng, Zhichao Lei, Jie Deng, Wenhuan Wei, Adwin Jahn, Qiuyu Chen, Xu Liu, Wei Huang, Junchao Yang, Tao Liu, Sheng-Ting Shen, Yen-Shuo Lin, Shih-Gu Huang, Hung-Yu Wei and Qiyue Li for their help and discussions during my PhD study in this group.

I also thank all the friends I have met in Seattle. You have not only given me support but also enriched my life of studying abroad in the beautiful Emerald City. I will remember the laughter every time we got together.

Last but not least, I am grateful to my dearest family: my father and my mother, for their endless love, support and encouragement for everything I do. They are the ones who make me strong and confident to conquer the hurdles set in front of me. This dissertation is dedicated to them.

Chapter 1 - Introduction

1.1. Motivations and Objectives

Fish abundance estimation is critically required for the commercially important fish populations in fisheries science. To address these needs and improve the quality of abundance survey, remotely-operated vehicle (ROV) based video is used to track and count live fish in underwater scenarios. However, video-based sampling for fish abundance estimates generates vast amounts of data, which present challenges to data analyses. The same situation occurs for on board chute based image data. These challenges can be overcome by using automatic vision analysis techniques for underwater fish tracking and counting, also for on board species identification, segmentation and length measurement. An example of the data collection and environment is shown in Figure 1.1.

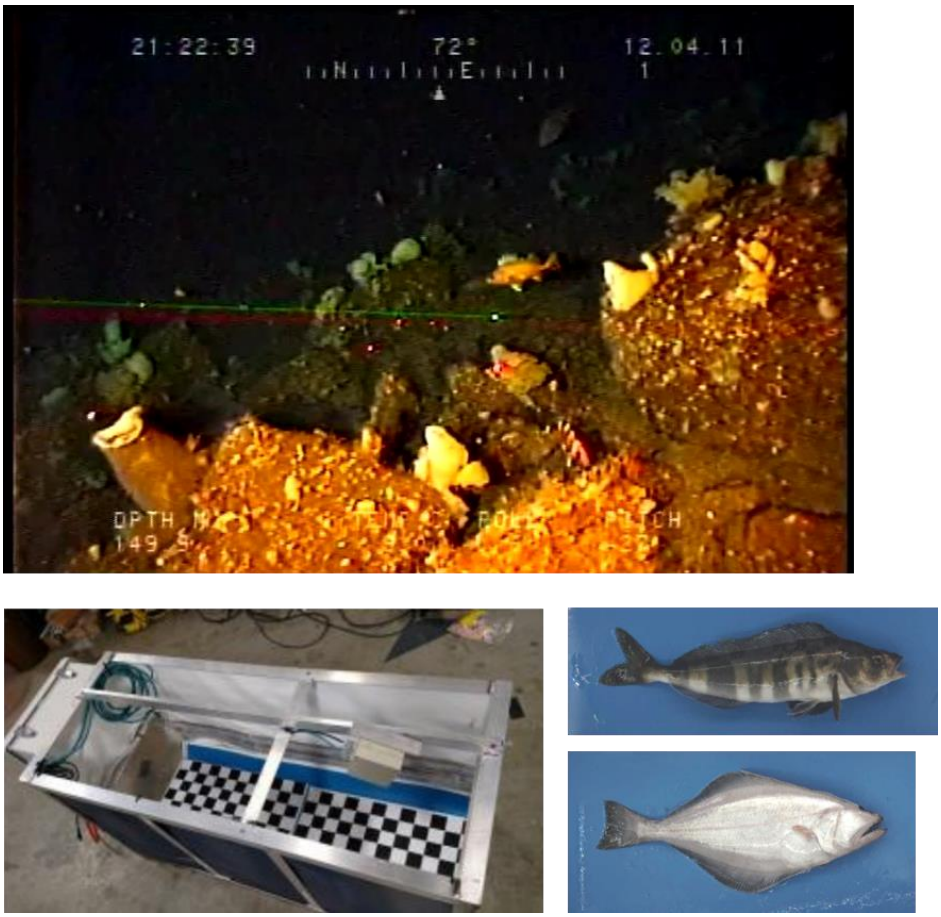


Figure 1.1: Environment and data collection.

1.2. Overview of Vision based Analysis in Fishery Applications

1.2.1. Species Identification

Fine-grained species identification has been a challenging task in recent year. As shown in Figure 1.2, even for humans, it is very difficult to distinguish the two salmon species. With only a limited small amount of training data, an efficient augmentation approach is proposed to prevent the over-fitting issue.

In the fishery survey, we can obtain a large amount of unlabeled data each year. At the first beginning, we may need human effort to label some data for training the species classifier. However, when new data comes in the next year, the performance of the trained classifier on the new dataset usually drops due to different data distributions, different camera distortions and different lighting conditions, as shown in Figure 1.3 and Figure 1.4. To overcome the problem with environment change between different dataset in the species identification, active learning approach is adopted for efficient labeling and model adaptation. We present a novel batch mode approach [2], [124] that combines the information given by an initially trained classifier and the data structure of unlabeled samples via sparse modeling based on uncertainty sampling. We discuss the advantages of our proposed method as follows.

- **Represent sample uncertainty via Gaussian kernels**. In the sample selection, we use sparse linear combination of Gaussian kernels to represent the uncertainty scores of unlabeled samples. The advantage of the linear combination has two folds. 1) Regions with high density and uncertainty in the pool data are encouraged to be represented by Gaussian kernel first since such regions have large impact on the representation error. 2) Since each Gaussian kernel can cover the uncertainty of a cluster of samples with high similarity, the algorithm will select samples with little overlapping information. In other words, diverse samples are selected at one time.
- **Selective sampling** [19]. We combine selective sampling before the optimization. The samples with low uncertainty in locality are filtered out. There are two advantages about this selective sampling strategy. On one side, the sparse representation is no longer

influenced by low uncertainty samples. As a result, the representation error can be largely reduced. On the other side, the number of samples is reduced dramatically, which results in faster convergence in the optimization.

- **Efficient optimization by approximated approaches**. We propose two approximated approaches two solve the sparse problem. The first one is greedy search method. In batch mode active learning, the classifier is updated after a batch of sample are selected and labeled. In the proposed greedy method, we process the sample selection sequentially without changing the classifier until a batch number of samples are selected. For each selection, we use Gaussian kernel to refine the uncertainty scores of neighbouring samples of the selected sample. Although the optimal solution is not guaranteed, the promising results can be achieved efficiently. The second approach is converting the problem into quadratic programming with relaxation to a l_1-norm constraint problem. In addition, the number of batch samples is taken into the consideration during the optimization. For the second approach, optimal solution is guaranteed for the l_1-norm constraint problem due to the convexity of the formulation.

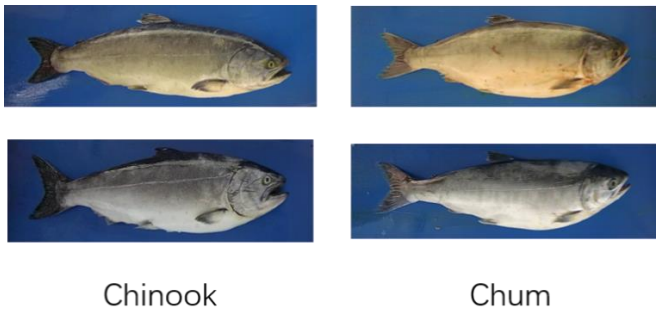


Figure 1.2: Two types of salmon species.

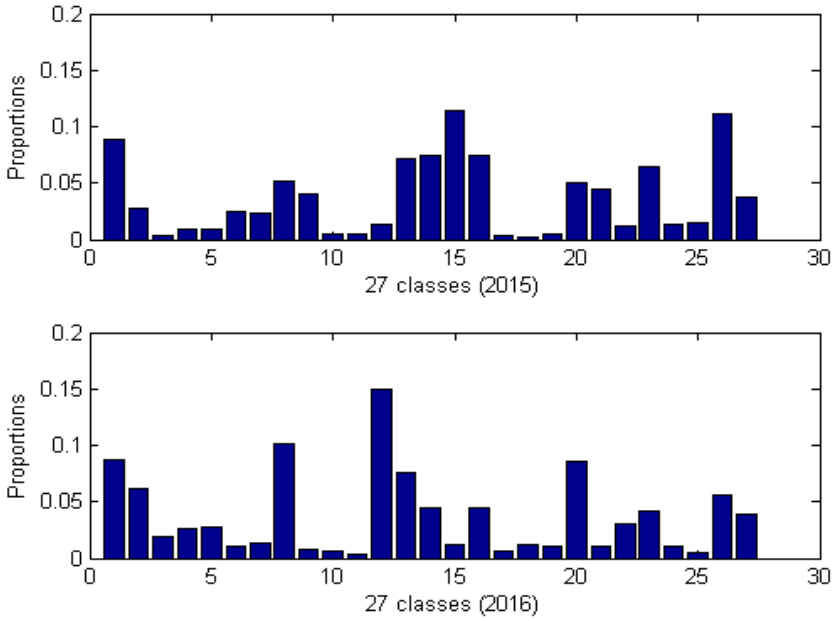


Figure 1.3: Data distributions of fish species in different years.

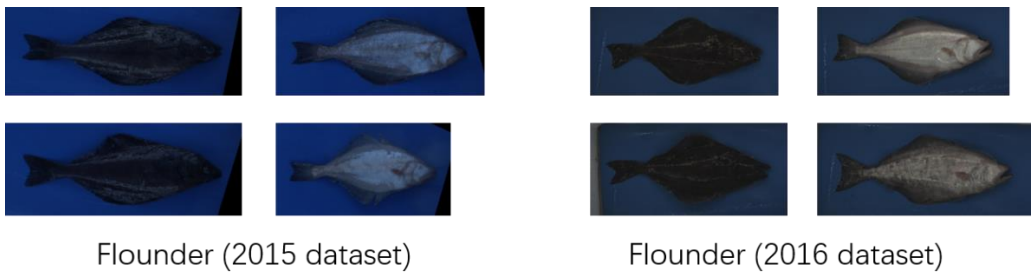


Figure 1.4: The captured images for flounder species in different years.

1.2.2. Segmentation Refinement

As for chute based fish segmentation, a coarse-to-fine contour-based method [122], [123] is developed for water drop and missing body recovery, with examples shown in Figure 1.5. First, several segmentation approaches [10], [11], [12], [13] are applied to the raw fish images to get the initial segmentation mask which is treated as the input of our developed refinement system. At the beginning, the initial segmentation contour is aligned with pre-trained shape models via an affine transform, this constitutes the coarsest level for entire contour alignment. Then the contour segments are refined iteratively to represent the poorly segmented or shape missing testing image. From coarse to fine, the segmentation refinement focuses more on the local part of the fish, which

allows more variation of the fish shape. The proposed method has three major contributions: 1) the refinement is processed from coarse to fine level; 2) the segmentation refinement is robust for any kind of initial segmentation; 3) segmentation can be also estimated with water drops and part of the object is out of the camera view.

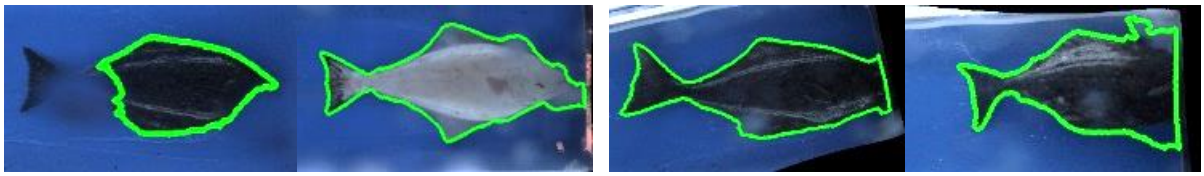


Figure 1.5: Examples of the challenges in the segmentation. Left two: images with water drops. Right two: images with part of fish out of the camera view.

1.2.3. Underwater Fish Tracking

Automatically tracking and counting live fish from remotely-operated-vehicle (ROV)-based underwater video data plays an important role in fish abundance estimation. Tracking algorithms facilitated by the tracking-by-detection framework have been widely investigated recently [3], [4], [5], [6], [7], [125]. In this tracking framework, the objects of interests are systematically tracked based on an online detector learned by previous frames. Different from their tracking procedure, since new objects may enter the field of view (FOV) at any time for our underwater environment, a pre-trained fish detector needs to be applied frame-by-frame to locate new objects. However, there are two main challenges in developing the fish tracking and counting system, i.e., how to decrease high false positive rate and false negative rate due to unreliable detections. The diversity of fish poses, the deformation of fish body shape and the color similarity between fish and background make a huge contribution to unreliable detections, resulting in large tracking and counting errors. Some examples are shown in Figure 1.6. Our proposed method specifically addresses unreliable detection issue with tracklet classifier and TrackletNet tracker (TNT) [66]. First, we define a graph model which treats each tracklet as a vertex. The tracklets are generated by appearance similarity with convolutional neural network (CNN) based features and intersection-over-union (IOU) between adjacent frames. Then the temporal locality-constrained linear coding (TLLC) [108] is adopted to represent each tracklet by a learned codebook.

Afterwards, an SVM classifier is used to remove the false-positive tracklets as the data cleaning step. Then, for every pair of two remaining tracklets, the similarity is measured by our designed multi-scale TrackletNet. Finally, the tracklets are clustered [65] into groups which represent individual object IDs. Our proposed tracking method has the ability to handle most of the challenges in underwater environment, and achieve promising results on fish tracking datasets compared with other state-of-the-art methods. With the proposed tracking method, the accuracy of tracking and counting is greatly increased, and the system is also accomplished in a fully automatic way.

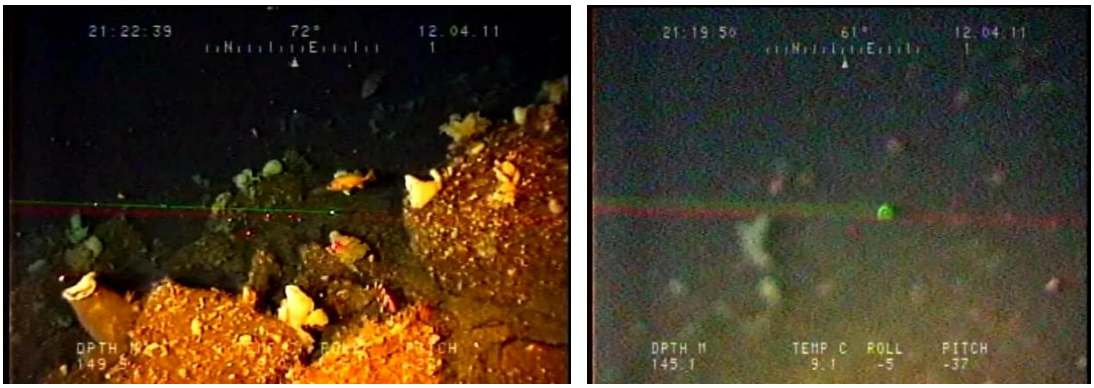


Figure 1.6: Examples of video data for fish abundance estimation.

Combining all the applications above, the proposed system can automatically process the input data, analyzing each individual fish observation, and perform fish abundance estimation in various kinds of marine habitats for either commercial, regulatory, or research purposes.

1.3. Contributions

1. Propose a novel augmentation technique for convolutional neural networks (CNN) based classifier in species identification.

2. Formulate the sample selection method as a sparse modeling problem and represent the unlabeled data by a linear combination of Gaussian kernels.

3. Combine uncertainty, diversity and density simultaneously in the sample selection in the active learning.

4. Approximate the sparse modeling problem by greedy search and quadratic programming, which can efficiently solve the problem.

5. Address the water drop and missing part issues efficiently in fish length measurement.

6. Propose a coarse-to-fine refinement approach for an arbitrary segmentation.

7. Propose a temporal pooling strategy for tracklet based feature extraction and classification.

8. Design a CNN based architecture, TNT, for similarity measurement between tracklets.

9. Transform the tracking problem into a tracklet based graph partition problem.

10. Solve the graph partition problem by an efficient clustering method.

1.4. Dissertation Organization

The rest of this dissertation is organized as follows.

Chapter 2: the background is briefly introduced, followed by a review of related work on fish species identification, active learning, segmentation refinement for length measurement and underwater fish tracking and counting.

Chapter 3: the species identification approach with effective augmentations is described. To adaptively improve the performance of the classifier given more and more unlabeled data, a novel active learning algorithm is proposed to solve an optimization problem which combines diversity, density and uncertainty into a unified framework. Extensive experiments and analysis of the proposed framework are given at the end.

Chapter 4: the proposed segmentation refinement is presented for species length measurement designed for chute based image data. A coarse-to-fine multi-scale approach is proposed for contour alignment with trained fish shape models. Extensive experiments for both qualitative and quantitative analysis are provided.

Chapter 5: the underwater fish tracking framework is introduced. A temporal locality linear coding is adopted for tracklet generation and classification. Then a deep neural network based tracker, TrackletNet tracker (TNT), is designed for similarity measurement between tracklets. Finally, a tracklet based graph model is built and clustering is conducted to obtain each fish ID.

Chapter 6: a conclusion of this dissertation is given by summarizing the main contributions and discussing some extensions to this work that lead to potential research topics in the future.

Chapter 2 - Background and Related Work

2.1. Species Identification and Active Learning

While image classification has been well investigated in machine learning and computer vision communities, there exists some fundamental challenges for live fish identification purpose. As a case of fine-grained image classification, different species also share a strong visual similarity. Besides, for free-swimming live fish, samples from the same species may vary a lot due to pose changes. In addition, low image quality caused by fast attenuation of light in the water, poor control over illumination, the ubiquitous organic debris, all these factors contribute to the challenge of fish identification task.

Existing methods for object classification can be divided into two categories, namely the supervised and unsupervised methods, based on feature extraction techniques. Supervised feature extraction method focuses on finding most significant features by human knowledge, such as the shape of fish body, the aspect ratio, the contour feature and the texture of the body. For instance, shape and size features are effectively extracted to represent fish [67]. A supervised method by separating fish into different parts is adopted in [68]. Histograms of oriented gradients (HOG) [69] of body parts and Fourier descriptors of the contour shape have been also used to describe fish features. Similarly, a contour matching technique is adopted in [70] for fish recognition. While good performances are shown in their results, some fundamental challenges remain existent for supervised methods. First, they rely on robust segmentation methods so that fish body parts can be located correctly easily. Second, occlusion needs to be carefully handled, otherwise the extracted contour of an occlude fish may seem meaningless. Third, recognition performance degrades when facing a huge diversity of fish poses. On the other hand, unsupervised methods learn informative features directly from images without the help of human knowledge. Some unsupervised feature extraction approaches for fine-grained recognition can be found in [71], [72], [73], [74], [75]. Kernel descriptors based on shape, texture, and color information have been constructed for template learning [71]. [72] localized distinctive details by roughly aligning the object parts. [73] learned deformable part models for fine-grained recognition. [74] used learned part models to

represent object parts for classification. [75] adopted fast R-CNN to detect and recognize underwater species

Active learning shows great power of improving the robustness of classifiers when dealing with limited training data or even without any ground truth labels. Representativeness of data have been studied in the sample selection strategy when no ground truth labels are given. As it is important to exploit the data distribution when selecting the data to be labeled [20], representativeness sampling tries to select the most representative data points according to the distribution of unlabeled data. Some well-known approaches of representativeness sampling include [21], [22], [23], [24], [25]. In [24], a simple concept, transductive experimental design, is proposed to explore available unlabeled data. In [21], the most representative points to reconstruct the whole dataset are selected in active learning by the Locally Linear Reconstruction (LLR) algorithm. Similarly, in [22], sparsity is taken into consideration in the reconstruction scheme for the sample selection. More recently, locality information by neighborhood samples is utilized in the reconstruction in [23]. However, for representativeness sampling based active learning, since no ground truth label information is given in the experiment setting, the sample selection is purely processed in an unsupervised way. The sampling strategy becomes inefficient if some assumptions in the unsupervised learning fail.

On the other hand, some active learning methods take advantage of a set of seed labeled samples to initialize the classifiers [26], such as uncertainty sampling, query-by-committee, expected model change and expected error reduction. Uncertainty sampling is a good way to utilize a pre-trained model in the sample selection. For example, for binary problems, feature points that are close to the classification boundary are chosen to label as the most uncertain samples [19], [27], [29], [30] based on different types of classifiers, like neural networks [19], [27] and support vector machine (SVM) [29], [30], [31], [32]. For multi-class classification problems, the first two most likely predictions are used to calculate the uncertainty [33], [34]. However, the performance of such uncertainty sampling based active learning largely depends on the robustness of the pre-trained classifiers. Sometimes uncertainty sampling even works worse than random sampling in

some applied settings when very limited labeled data are used to train the initial classifier [35], [36], [37].

Since the representativeness of the unlabeled data can be also utilized with a pre-trained classifier, many recent approaches have incorporated the representativeness in their uncertainty design to overcome the weakness of uncertainty sampling based methods [38], [39], [40], [41]. In [38], the distribution of the data is taken into consideration in the sample selection. Diversity is incorporated in the version space reduction in [39]. In [41], a convex optimization framework is proposed with diversity incorporated in active learning with an arbitrary classifier.

The most recent and related work with diversity maximization in the sample selection is proposed in [40], where the sample selection is modeled as an optimization problem with the following formulation,

$$\hat{f} = \text{argmin}_f - f^T a + \frac{1}{2} f^T K f,$$
$$\text{s.t.} \sum_{i=1}^{n} f_i = 1, f_i \geq 0, \tag{2.1}$$

where f is the updated ranking score; vector a is the uncertainty scores of unlabeled samples; K is a kernel matrix with $K_{i,j} = \exp(-\frac{\|x_i - x_j\|^2}{\sigma^2})$ which measures the similarity between points x_i and x_j. The first term $-f^T a$ penalizes less if samples with high uncertainty also get higher ranking scores. With the kernel matrix K in the second term $f^T K f$, the algorithm tends to give higher ranking scores to samples with less similarity. The problem is optimized to find the best trade-off between the uncertainty and the diversity. However, there are two major weakness in the algorithm: 1) The formulation always selects isolated distinct samples with high uncertainty first. This is because isolated samples are dissimilar to other samples which will generate little penalty on the second term. However, since isolated samples are far away from the data density, these samples are "unimportant". Selecting such samples is not very helpful for the classifier to improve the performance. This results in inefficient selection especially when we are interested in selecting a small batch of samples. 2) The algorithm does not take the batch size in the optimization. In fact, the batch size does matter in the sample selection. Take an extreme situation for example. If the

batch size is 1, then the sample that lies in the center of the pool data would be the most representative sample. However, if the batch size is 2, then we may divide the pool data into two clusters and the sample near the center of each cluster would be the most representative sample. In other words, the ranking order varies with the batch size.

2.2. Segmentation Refinement

Measuring the fish size and length requires a robust segmentation approach. Depend on whether requiring training data, the segmentation methods can be roughly divided into two major categories, i.e., unsupervised approaches [42], [43], [44], [45], [46], [76], [77] and supervised approaches [12], [13], [78], [79], [49], [50], [51], [80].

The advantages of unsupervised approaches are obvious. First, it does not need any human effort to annotate the ground truth labels for each image pixel. Second, it can easily segment a new class object that is not in the training data labels. The drawbacks are also obvious. Without the supervision of the ground truth, such methods cannot achieve very good performance. Most unsupervised segmentation methods adopt clustering approaches to cluster pixels into different groups. For example, Achanta et al. [42] propose a simple linear iterative clustering (SLIC) approach, which adapts a k-means clustering to efficiently generate superpixels. However, this method can only be used to generate superpixels, but not to segment the entire object. Graph-cut based approaches [43], [44], [76], [77] use graph model to minimize the energy on the edges to perform the segmentation. However, graph-based methods usually require a lot of computation time, in particular for images with high resolution. Moreover, the mean-shift method [45] is also utilized to cluster pixels of the image, but it is sensitive to the bandwidth of the mean-shift kernels. While Arbelaez et al. [46] reduce the problem of image segmentation to contour detection, the segmented objects are not class specific and it is not clear how to extract the foreground object out of the image. Although these unsupervised methods are efficient even without training, they are not suitable for our tasks. They are still easily affected by water drops and do not address the situation when the part of fish is out of the field of the camera.

With more and more labeled datasets [81], [82] available in recent years, supervised learning-based approaches become more powerful in dealing with segmentation. With the supervision of the ground truth labels, more complex models can be trained to achieve higher segmentation accuracy, like neural network-based approaches [12], [13], [78], [79], [49], [50], [26], [80]. Long et al. [12], [13] adopt fully convolutional networks (FCNs) for image segmentation, which can take an input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. However, this approach treats the segmentation as pixel classification without any other constraints in the cost function, resulting in non-sharp boundaries and blob-like shapes. In [79], Zheng et al. combine the strengths of FCN and Conditional Random Fields (CRFs)- based probabilistic graphical modeling and formulate meanfield approximate inference for the Conditional Random Fields with Gaussian pairwise potentials as Recurrent Neural Networks. Similarly, Chen et al. [49] combine deep networks with fully connected CRFs to perform the segmentation. Combining CRF can recover detailed information near the boundaries, which achieves better performance. However, FCN-based architectures still cannot perform as good as deconvolutional networks [80]. Badrinarayanan et al. [50] propose encoder-decoder architecture networks, in which the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. This eliminates the need for learning to upsample in the deconvolution. Although neural network-based methods show promising results, they rely on large training data and cannot segment a new class object that is not in the training ground truth labels. Still, such methods are not suitable for our task, which require large dataset for training, cannot better utilize the prior knowledge of fish shapes, and cannot well segment blur regions and recover the missing fish bodies.

Apart from the general segmentation methods, there are also some methods that specifically deal with water drop segmentations and fish body segmentations. Some works [83], [84], [85], [86], [87] have been done for water drop detections or blur detections. For example, Alippi et al. [83] propose a method which detects external disturbances on camera lens by comparing the blur measures of a series of frames which contain the same scene acquired from a static camera.

Kanchev et al. [84] propose an algorithm for detecting blurred regions in images by using wavelet-based histograms and SVM. Blur measure is defined on the fish contour to locate the water drop region by Huang et al. [85]. Moreover, Liu et al. [86] develop several blur features to detect blur regions for general images. However, such water drop detection methods do not illustrate how to deal with the issue in terms of segmentation. Chan et al. [87] propose a two-stage image segmentation method for blurry images with Poisson or multiplicative gamma noise. However, the water drop issue in our fish length measurement only occurs locally, which is not suitable for their setting. As for fish body segmentation, Chuang et al. [11] propose double local thresholding (DLT) method which uses the color histogram to distinguish foreground and background. Similarly, Huang et al. [85] combine the DLT and Gaussian mixture model (GMM) [10] to model the static background to help extract the foreground fish object. However, these two methods purely rely on unsupervised approaches and do not take advantage of fish shape priors, which still cannot recover the missing fish bodies.

To better address the segmentation challenges, the prior knowledge of the fish shape needs to be exploited in the segmentation. If we have a shape contour model with pose variations, the water drop region can be estimated and predicted with the help of other parts of the fish body based on the shape contours. Similarly, when dealing with the missing shape issue where part of the fish is out of the camera view, we can also efficiently predict the missing part with the help of the rest of the fish body based on the contour model even with very limited training data.

2.3. Underwater Fish Tracking

Visual tracking has been an active research topic since the early 1980s and keeps advancing both in theory and practice. Followed by the tracking-by-detection framework [3], [4], [5], [6], [7], [28], the objects of interests are systematically tracked based on an online detector learned by previous frames. For example, an adaboost classifier is utilized in [4] to extract the foreground object from the background. Multiple particles around the object and the properties of low-rank matrix are effectively utilized in moving object tracking [3]. Based on the kernelized correlation filter (KCF) [7], the property of a circulant matrix is adopted to accelerate the speed for real-time

tracking. They all assume that no new objects of interests will appear in the middle of the video sequence. However, this assumption cannot fit for our system since new fish may enter the field of view at any time during the video. With the help of pretrained detectors, multiple kernels based tracking [9] has been reported to achieve good performance. However, the tracking is heavily relied on the assumption of reliable detections. Without reliable detection, the tracking and counting error would increase significantly. Moreover, the tracked 2-D objects can be projected onto 3-D world by taking advantage of the estimated ground plane [14]. However, there is no ground plane for underwater videos. As a result, it is not likely to easily implement fish tracking in 3-D.

There are also some automatic systems developed for fish detection and tracking [8], [15], [16], [17], [18]. Specifically, in [15] and [16], Gaussian mixture modeling (GMM) and a moving average algorithm are combined for detection. But there are several drawbacks: 1) it only can be used for static cameras; 2) it cannot detect static fish; and 3) it is hard to tune the thresholds for different kinds of underwater scenarios. In [17], Gabor filters and projection curve segmentation are used to find the locations of the fish. The drawbacks are also obvious: 1) the Gabor-filter based detection cannot deal with different fish poses; 2) the segmentation cannot work well for different orientations of fish; and 3) the method cannot deal with multiple fish in one frame. Moreover, the shape-based level set [18] is applied to detect fish. However, it cannot deal with fish with twisted body shape.

In addition to visual tracking and fish tracking, there are also general approaches for multi-object tracking (MOT). We introduce MOT based on the following categories.

Graph Model based Tracking. Most of the recent multi-object tracking approaches are based on tracking-by-detection schemes [88], [89], which associate given detections across frames and estimate object locations when missing detection or occlusion occurs. Many tracking methods are based on graph models [90], [91], [65], [92], [93], [94], [95], [96], [97] and solve the tracking problem by minimizing the total cost. Generally, there are two categories of graph models. One treats the individual detections as the vertices [90], [91], [92], [94], while the other using tracklets

as vertices [65], [95], [96], [97]. For detection-based graph models, there are two major disadvantages. First, one of the important assumptions in graph models is the conditional independence of the vertices. However, detections are not conditional independent from frame to frame if we want to track an object in a long run. Hence, the temporal information is not well utilized. Second, a detection-based graph usually comes with a very high-dimensional affinity matrix, which is very time consuming to find a good solution in the optimization. On the contrary, tracklet-based graph models can better utilize the information from a short trajectory to measure the relationship between vertices, but the mis-association should be carefully handled in the tracklet generation step.

Recurrent Neural Networks (RNNs) based Tracking. Besides graph models, recurrent neural networks (RNNs) based tracking also plays an important role in recent years [98], [99], [100], [101], [102]. For example, [102] first time proposes an end-to-end learning approach which uses an RNN to model the target motion. However, drawbacks are also obvious, i.e., it can be easily affected by the camera motion and does not utilize appearance information in the association. Similarly, [101] minimizes the regression error and association error in a unified framework by using long short term memory (LSTM) blocks. One advantage of RNN-based tracking is the ability of online prediction. However, along with the propagation of RNN blocks, the correlation between two far-away detections becomes very weak, especially for high-dimensional appearance features [100]. As a result, the drift error can be easily accumulated during the long-time occlusion. The performance of RNN-based methods degrades in the long run and sometimes can be easily affected by unreliable detections.

Feature Fusion based Tracking. Features are very important in the tracking-by-detection framework. There are two types of features that are commonly used, i.e., appearance features and temporal features. For appearance features, many works adopt CNN-based features and treat tracking as a reidentification (Re-ID) task [103], [64], [90], [104]. For example, [103] proposes an adaptive weighted triplet loss for training and a novel technique for hard-identity mining. [64] adopts the reranking technique [104] in calculating the feature similarity. Besides CNN-based

features, histogram-based features, like color histograms, histogram of oriented gradients (HOG), and local binary patterns (LBP), are still effective if no labelled training data are available [65]. For temporal features, the location, size, and motion of bounding boxes are commonly used. Given the appearance features and temporal features, the tracker can fuse them together, like [64], [91], [65]. However, it is still empirical and difficult to determine the weights of different types of features.

End-to-End based Tracking. Another category of tracking is based on end-to-end frameworks [105], [106], [107], where we input raw video sequences and output object trajectories. In other words, the detection and tracking are trained jointly in a single-stage network. One major advantage of this framework is that the errors will not be accumulated from detection to tracking. The temporal information across frames can help improve the detection performance, while reliable detections can also further improve the tracking performance. However, such a framework requires a lot of training data with a large diversity of different scenarios. Without enough training data, over-fitting becomes a severe problem. Unlike detection based training, annotations of tracking for video sequences are usually hard to be obtained, which become the major limitation of the end-to-end tracking framework.

Chapter 3 – Species Identification

In this section, we demonstrate how to deal with fine-grained species identification with the help of a CNN classifier. To address the issue with only a small amount of training data, we propose a novel augmentation technique that can better utilize the limited data in the training.

In real-world applications, dataset is obtained from year to year. We also describe how to use active learning approach to improve the designed classifier for species identification.

3.1. Fine-Grained Classification with Cut-off Augmentation

With the fast development of machine learning and computation capacity, convolutional neural networks (CNN) has shown great power in recent years. In this section, how to take advantage of CNN architecture with efficient augmentation techniques is described in detail to deal with fine-grained classification problem.

3.1.1. CNN Architecture

Here we adopt Inception-Resnet-V2 [60] as our architecture for classification, which takes advantage of both Inception block [61] and Resnet architecture [62]. The architecture is shown in Figure 3.1.

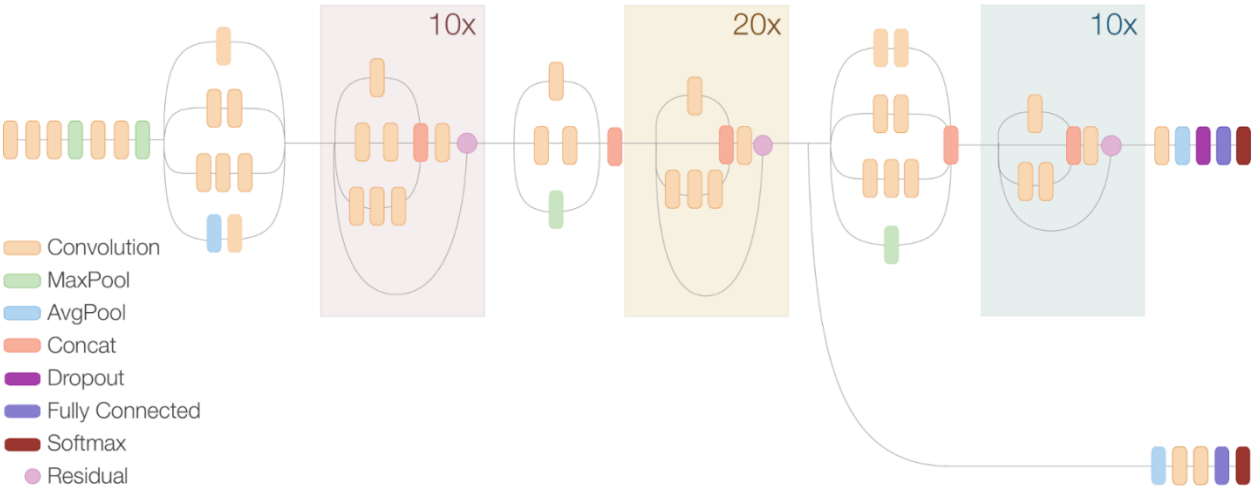


Figure 3.1: Inception-Resnet-V2 architecture.

3.1.2. Augmentation

To address the issue that only a small amount of training data is available, we use augmentation techniques to prevent over-fitting problem. In addition to traditional ways of augmentation, such as rotation, flip, color distortion, we also add cut-off as one type of augmentation, as shown in Figure 3.2. In the cut-off augmentation, we randomly select two points from two boundaries of the image. Then we cut off the smaller part along the line, as shown in Figure 3.3.

There are two major advantages of the cut-off augmentation, i.e., 1) the neural networks can learn each individual part efficiently by input hard examples with the cut-offs, 2) it can also solve the identification problem when portions of fish are out of the camera view.

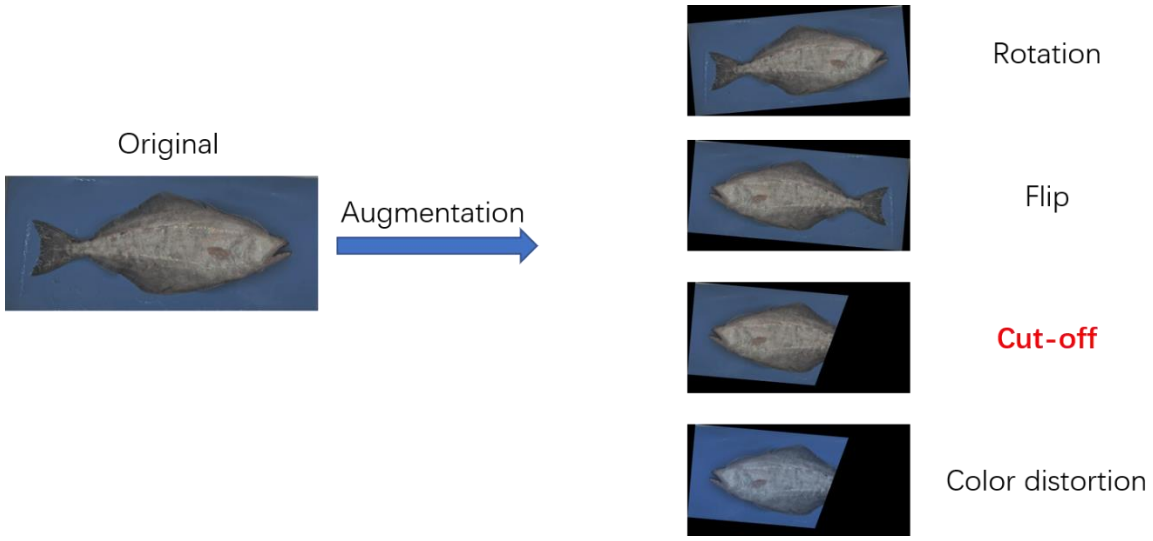


Figure 3.2: Augmentation for training.

Figure 3.3: Cut-off augmentation.

3.1.3. Experimental Results and Analysis

1) Datasets

A) Fish 2016 Dataset

In this dataset, it contains 6,740 images and 43 species. We split the dataset into training and testing, with 6,042 and 698 images, respectively.

B) Salmon Dataset

The dataset contains 217 salmon images with two fine-grained classes, i.e., Chinook and Chum. We split the dataset into training and testing, with 162 and 55 images, respectively. Examples of salmon species are shown in Figure 3.4. As we can see, color information is not a primary feature for the identification. Same type of salmon may have different dominant colors and different salmons may share the similar color. Moreover, the lighting conditions change a lot across different years, which make the identification more difficult. Sometimes the salmons are not distinguishable from images by humans except you are an expert.

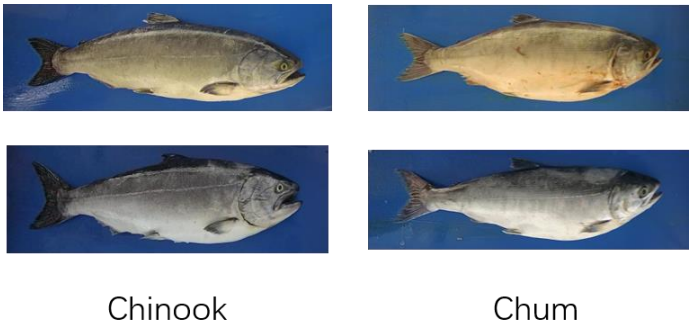


Figure 3.4: Examples of salmon species.

2) Classification Performance

The results on fish 2016 dataset are shown in Table 3.1. We can see that the method with the cut-off augmentation achieves the best performance.

Table 3.1: The classification performance on fish 2016 dataset

Method	**Acc**
BoF (w/o multi-spectral) [63]	94.0%
BoF (w multi-spectral) [63]	94.4%
Inception-resnet-v2 (w/o cut-off aug, w/o cut-off in testing)	93.1%

Inception-resnet-v2 (w/o cut-off aug, cut-off in testing)	76.3%
Inception-resnet-v2 (w cut-off aug, cut-off in testing)	93.1%
Inception-resnet-v2 (w cut-off aug, w/o cut-off in testing)	**94.5%**

For the salmon dataset, we crop the tail part of the image for classification since the black spots on the fish tail is a significant feature for identifying chum and chinook species, which can be shown in Figure 3.5.

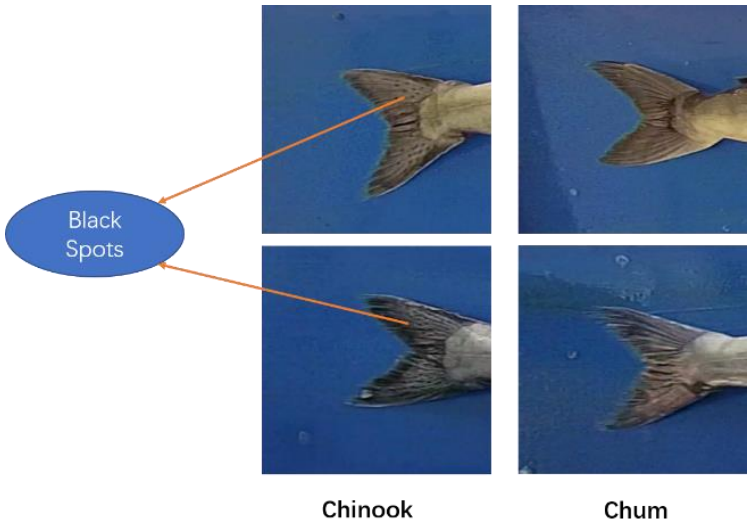


Figure 3.5: Black spots on chinook salmon compared with chum.

After training, the confusion matrix of the testing data is shown below, which has 98.2% accuracy.

Table 3.2: The confusion matrix of the salmon dataset

	Chinook	Chum
Predicted Chinook	39	0
Predicted Chum	1	15

3.2. Active Learning via Sparse Modeling

The flowchart of our proposed framework is shown in Figure 3.6. First, a classifier is initially trained on the labeled set at the beginning. Then the unlabeled data is tested by the trained classifier. Based on the predictions, uncertainty scores are obtained to measure the ambiguity of the unlabeled

data. Then the sparse modeling is used for sample selection. After sample selection, new labeled samples are added to the training dataset to re-train the classifier. Finally, the performance of active learning is evaluated on an independent testing dataset. In this section, we will introduce the SVM classifier, uncertainty measure design, sample selection via sparse modeling and an approximated solution to the sparse problem. Note that, in this paper we only use SVM classifiers for our active learning due to the much lower computational complexity requirement, compared to most recent high computational demanding convolution neural networks (CNNs), since the development environment needs to be deployed on boat for continuous adaptive learning. In fact, the proposed scheme can also be used in many types of classifiers, such as CNNs, if the complexity requirements can be relaxed.

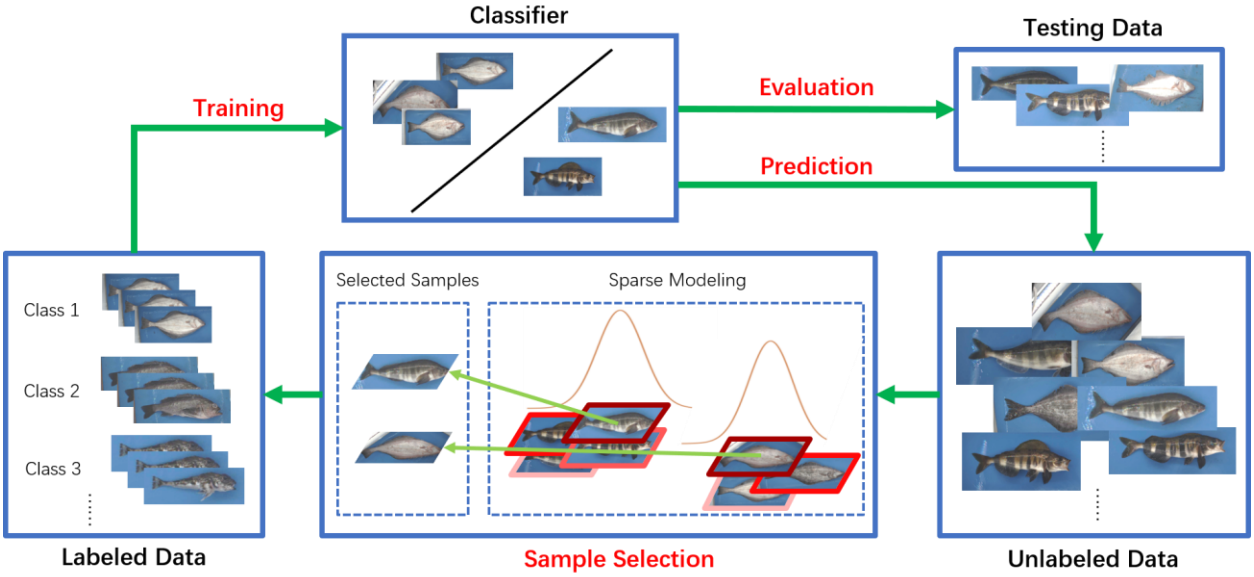


Figure 3.6: The flowchart of the learning system.

3.2.1. Multi-Class SVM Overview

For a multi-class classification problem, we can train linear SVM [53] classifiers based on the "one vs. the rest" strategy. Assume we have K classes, and for the k-th class, we treat the training samples that belong to this class as positive samples and all the rest samples not in this class as negative samples. Then the k-th classifier is trained based on the following equation provided in [24],

$$\widehat{w}_k = \arg\min_{w_k \in R^d} C_p \sum_{i=1}^{N} l_2(y_i w_k^T x_i) + \frac{1}{2}\|w_k\|^2, \quad (3.1)$$

where l_2(z) is given by $l_2(z) = \max(0, 1-z)^2$, $\widehat{w}_k$ are learned weights for the k-th classifier, C_p is a real-valued regularization parameter, and (x_i, y_i) is the *i*-th instance-label pair. We use l_2 loss instead of hinge-loss to make the training more efficient since the gradient of l_2 loss is continuous. For simplification, we use $\|\cdot\|$ without subscript to denote the l_2 norm $\|\cdot\|_2$. The final SVM classification result can thus be determined by the following equation,

$$\hat{k} = \arg\max_{k \in \{1,2,\ldots,K\}} (\widehat{w}_k^T x_i), \quad (3.2)$$

where $\widehat{w}_k^T x_i$ is the prediction score of the testing sample x_i corresponding to the *k*-th class, and it represents the distance from the feature point x_i to the decision hyper-plane of the *k*-th class.

3.2.2. Uncertainty Measure Design

In active learning, uncertainty sampling aims to choose the most uncertain samples from the unlabeled data pool to label. For SVM based classifier, it is common to use the distance between the first two most likely predictions. Similar to [33], [34], we define the uncertainty score based on the Best vs. the Second Best (BvSB) strategy,

$$s_{\mathrm{BvSB}}(x_i) = \max(\widehat{w}_{k_2}^T x_i - \widehat{w}_{k_1}^T x_i + 1, 0), \quad (3.3)$$

where k_1 and k_2 are the first two most likely predicted classes. The uncertainty score is restricted to [0, 1]. With smaller $s_{\mathrm{BvSB}}(x_i)$, x_i has higher confidence in k_1 class. Therefore, $s_{\mathrm{BvSB}}(x_i)$ reflects an uncertainty measure of the testing point x_i.

3.2.3. Sample Selection via Sparse Modeling

Given uncertainty scores generated from the classifiers, we would like to select the most informative samples for a query. The simplest selection strategy is that we always select the samples up to the batch size, B_q, with the highest uncertainty scores. However, this strategy ignores the relations among the pooled unlabeled samples. Sometimes the samples with top uncertainty are very similar to each other. We should avoid selecting samples with redundant information in the same batch.

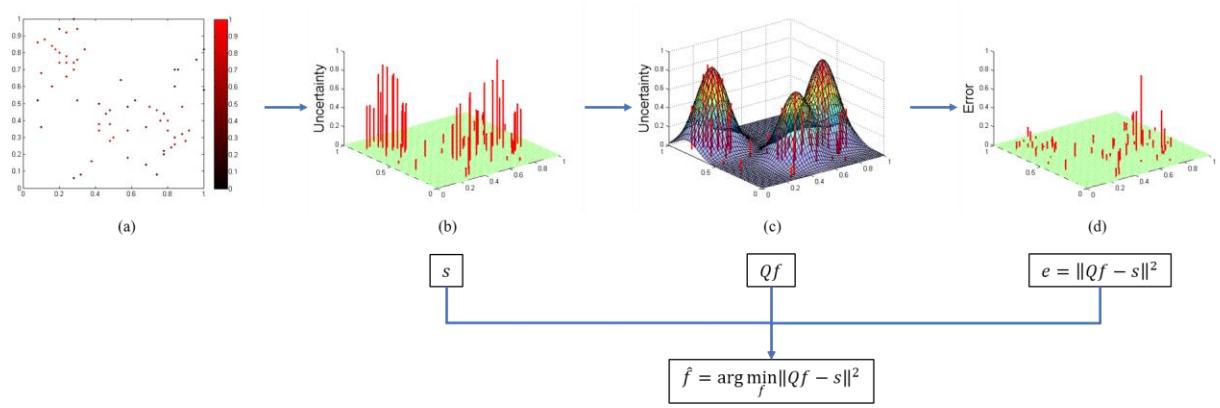


Figure 3.7: Overview of the sparse modeling for sample selection. (a): Uncertainty scores for feature points in 2-D space. The color from black to red represents the uncertainty score from low to high. (b): Uncertainty scores are represented in z-axis. (c): Use combination of selected Gaussian kernels to represent the uncertainty scores. f is a sparse vector in which only the indices of selected samples have non-zero values. Q is a collection of Gaussian kernels of all feature points. (d): Representation error with selected Gaussian kernels.

To achieve this goal, we can formulate the problem via sparse representation as shown in Figure 3.7. In other words, we want to select a few samples that can cover the information of the pool data as much as possible. To be specific, we propose the following formulation to modify the uncertainty scores before sample selection,

$$\hat{f} = \arg\min_f \|Qf - s\|^2,$$
$$s.t. \|f\|_0 = B_q,\ \mathbf{0} \le f \le \mathbf{1}, \tag{3.4}$$

where s is the original uncertainty scores, $\hat{f}$ is the modified uncertainty scores, $\|f\|_0 = \text{card}(f)$ represents the number of non-zeros entries, $\mathbf{0}$ and $\mathbf{1}$ are all-zero vector and all-one vector, respectively, B_q is the batch size and Q is the similarity matrix among all the pooled samples. Specifically, $Q_{i,j}$ represents the similarity between samples i and j in the range of [0, 1]. The traditional way of designing similarity matrix Q is using the Gaussian kernel of the Euclidean distance between two points, i.e.,

$$Q_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right). \tag{3.5}$$

However, when dealing with high-dimensional data points, which are commonly very sparse, the Euclidean distance might not be a good choice to represent the similarity. To better represent the similarity between two samples, we define the matrix Q as

$$Q_{i,j} = \begin{cases} \exp\left(-\frac{\|\tilde{x}_i - \tilde{x}_j\|^2}{\sigma^2}\right), & \text{if } i \in N_j, \\ 0, & \text{if } i \notin N_j, \end{cases} \tag{3.6}$$

$$\tilde{x} = [\hat{w}_1, \hat{w}_2, \dots, \hat{w}_K]^T x, \tag{3.7}$$

where $\tilde{x}_i$ and $\tilde{x}_j$ are transformed data samples of the initial data x_i and x_j, N_j is the neighbor index set of the *j*-th sample. Here we use the learned weights of SVM classifiers as the data transform.

To better illustrate the formulation in Eq. (3.4), let us assume there are N_U unlabeled samples, we can write matrix Q as $Q = [q_1, q_2, \dots, q_{N_U}]$ and each column vector q_j denotes the similarity weights between the *j*-th sample and all unlabeled samples via the Gaussian kernel. In this formulation, we are interested in looking for a sparse linear combination of the similarity weight vectors centered at selected samples, i.e., $Q\hat{f}$, to represent the original uncertainty scores s. After the problem is solved, the indices of non-zero entries in $\hat{f}$ would be our selected samples.

3.2.4. Approximated Solution 1: Greedy Search

The solution to the problem in Eq. (3.4) can be well approximated using greedy search method, i.e., we can select samples one-by-one and modify the uncertainty scores after each selection. Note that this greedy search method still follows the batch mode setting since there is no need to update the classifier after each sequential selection. We denote the similarity matrix Q as $Q = [q_1, q_2, \dots, q_{N_U}]$, where each column vector q_j in Q represents Gaussian kernel weights centered at the location of $\tilde{x}_j$. For the *t*-th selection from 1 to B_q, the selection strategy is as follows,

$$\hat{k}^t, \hat{f}_{\hat{k}^t} = \arg\min_{j \in U, f_j} \left\| f_j q_j - s^t \right\|^2, \tag{3.8}$$

where s^t is the uncertainty scores of all unlabeled data at time t, f_j is a scalar which represents the modified uncertainty score of the j-th sample, U is the index set of unlabeled pool data, $\hat{k}^t$ is the sample index selected for a query and $\hat{f}_{\hat{k}^t}$ is the modified uncertainty score for the selected sample.

This can be solved by sequentially obtaining $\hat{k}^t$ and $\hat{f}_{\hat{k}^t}$ using

$$\hat{k}^t = \arg\max_{j \in U} q_j^T s^t \,, \tag{3.9}$$

$$\hat{f}_{\hat{k}^t} = \arg\min_{f_{\hat{k}^t}} \| f_{\hat{k}^t} q_{\hat{k}^t} - s^t \|^2. \tag{3.10}$$

In Eq. (3.9), the sample with the maximum correlation between the Gaussian kernel q_j and uncertainty score s is selected. Then the modified uncertainty of the selected sample is calculated from Eq. (3.10).

After each selection, the remaining uncertainty is calculated from

$$s^{t+1} = \max(s^t - \hat{f}_{\hat{k}^t} q_{\hat{k}^t}, 0). \tag{3.11}$$

For each iteration, we keep the uncertainty score s^{t+1} to be non-negative. Then we move $\hat{k}^t$ from the unlabeled set U to the labeled set L. This greedy search method is similar to orthogonal matching pursuit (OMP) [54], except that we only keep non-negative values for residuals in Eq. (3.11). The approach is summarized in Algorithm 1. We name this method as sparse modeling by greedy search (SMGS), which means using greedy search method to solve the sparse modeling problem.

Although the sparse modeling problem can be approximated using Algorithm 1, there are still three major drawbacks of the formulation in Eq. (3.4): 1) the sparse representation is sensitive to the samples with low uncertainty scores; 2) the uncertainty, diversity and density are not well combined in the formulation; 3) optimal solution is not guaranteed using greedy search method. We will illustrate how we can overcome these drawbacks in the following section.

3.2.5. Selective Sampling for Sparse Modeling

For multi-class classification problem, there is often the case that samples with high similarity may have a large difference in the uncertainty. This situation results from the non-robust classifier due to the limited training data. Therefore, the neighboring samples for a given selected sample may have large difference in the uncertainty. Once we apply a Gaussian similarity kernel on the given sample, the samples with high uncertainty cannot be well represented by the kernel if several low uncertainty samples are around. This is because the loss function defined in Eq. (3.4) is to minimize the representation error of all samples including low uncertainty samples as illustrated

in the example given in Figure 3.8. From the Figure 3.8-(c), we can see that low uncertainty samples can have a large effect on the representation error. With the selective sampling strategy in Figure 3.8-(d), the samples with low uncertainty are filtered out before the sparse modeling, resulting in small representation errors as shown in Figure 3.8-(f).

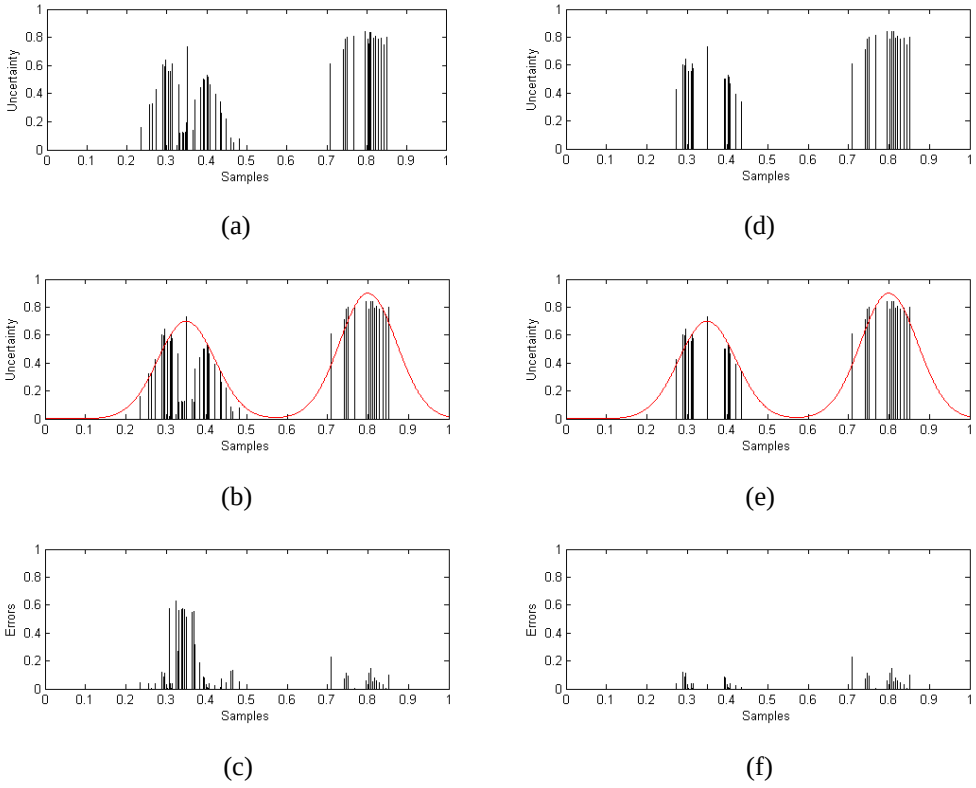


Figure 3.8: An example of selective sampling. (a) Uncertainty of all unlabeled samples. (b) Gaussian kernels for sparse modeling. (c) Representation errors via sparse modeling. (d) Uncertainty of pre-selected unlabeled samples. (e) Gaussian kernels for sparse modeling. (f) Representation errors via sparse modeling for pre-selected unlabeled samples.

Inspired from [19], we adopt selective sampling strategy to reduce the sampling space. We design a locality thresholding method to select high uncertainty samples among neighboring samples. Given an unlabeled sample j, we compare it with its neighboring unlabeled samples $i \in N_j$. We define the uncertainty influence, $I_{i,j}$, as a weighted uncertainty score from the sample i to the sample j, i.e., $I_{i,j} = Q_{i,j} s_i$. If $I_{i,j}$ has a much higher value than s_j, we should not select

the sample j since it has a much lower uncertainty than its neighboring samples. Let's define the influence difference d_j as,

$$\begin{aligned} d_j &= \max_{i \in N_j} I_{i,j} - s_j \\ &= \max_{i \in N_j} Q_{i,j} s_i - s_j. \end{aligned} \tag{3.12}$$

To be specific, the samples with $d_j > d_{\text{thresh}}$ are filtered out, where d_{thresh} is a pre-defined threshold. In other words, we select samples with the uncertainty scores not much lower than the uncertainty of the neighboring samples. We denote the index set of selected samples as S_U. Note that this pre-selection step filters out samples in their local neighbors rather than filtering all samples with a global threshold, which is more suitable for sparse representation.

There are two advantages about this selective sampling strategy. On one side, the sparse representation is no longer influenced by low uncertainty samples. On the other side, the number of candidate samples is largely reduced, which also reduces the complexity in the optimization.

3.2.6. Combine Diversity, Density and Uncertainty

In this subsection, we will demonstrate how we combine diversity, density and uncertainty by modified sparse modeling. After selective sampling, we only focus on a subset of the unlabeled samples, i.e., S_U. Hence, we modify the variables in Eq. (3.4) with $s = [s_{k_1}, s_{k_2}, \dots, s_{k_m}]^T$ and $Q = [q_{k_1}, q_{k_2}, \dots, q_{k_m}]$, where $k_1, k_2, \dots, k_m \in S_U$ and $m = \text{card}(S_U)$. Moreover, we rewrite Eq. (3.4) as,

$$\begin{aligned} \hat{f} &= \arg\min_f \|Qf - s\|^2, \\ &= \arg\min_f \tfrac{1}{2} f^T Q^T Q f - f^T Q^T s \\ s.t. &\ \|f\|_0 = B_q,\ \mathbf{0} \le f \le \mathbf{1}. \end{aligned} \tag{3.13}$$

We analyze the above formulation in three aspects as follows.

(1) Diverse term. The first term $\frac{1}{2} f^T Q^T Q f$ measures the diversity in the sample selection, where $A = Q^T Q$ is a positive semi-definite matrix with $A_{i,j}$ measuring the similarity between

samples i and j. We can see that if $A_{i,j}$ has a high value and both i and j have been selected, then $f_i A_{i,j} f_j$ would contribute a high loss. Since we want to minimize the loss function in Eq. (3.13), this term guarantees that samples with high similarities cannot be selected at the same time, i.e., diverse samples are encouraged to be selected.

(2) Density term. The second term, $-f^T Q^T s$, measures the density in the sample selection. Since f is a sparse vector, only selected samples can have non-zero entries. Moreover, we can treat Q as a Gaussian smoothing operator since the columns in Q are Gaussian kernels. Then Qf plays a role of convolution between a sparse vector and Gaussian kernels. This operation turns a sparse vector f into a dense vector. If we have a large density of samples around selected samples, then there would be a high correlation between Qf and s, which contributes a low loss in $-(Qf)^T s$.

(3) Uncertainty trade-off. To emphasize high uncertainty samples, an uncertainty term, $-f^T s$, can be added to strengthen the role of uncertainty in the sample selection as shown in Eq. (3.14).

We can relax the density term and uncertainty term with penalty parameters λ_1 and λ_2, so that the modified formulation of the sparse modeling becomes,

$$\begin{aligned} \hat{f} &= \arg\min_f \frac{1}{2} f^T Q^T Q f - \lambda_1 f^T Q^T s - \lambda_2 f^T s, \\ &= \arg\min_f \frac{1}{2} f^T Q^T Q f - f^T (\lambda_1 Q^T + \lambda_2 I) s, \\ &s.t. \|f\|_0 = B_q,\ \mathbf{0} \le f \le \mathbf{1}. \end{aligned} \tag{3.14}$$

To better demonstrate the difference between Eq. (3.14) and Eq. (2.1) [40], we make some detailed analyses as follows.

- **Density analysis**. Eq. (2.1) in [40] does not use the density term of Eq. (3.14), $-f^T Q^T s$. As a result, isolated distinct samples are encouraged to be selected in Eq. (2.1). This is because isolated samples are dissimilar to other samples, therefore Eq. (2.1) will generate little penalty on the diverse term. However, since isolated samples are far away from the data density, these samples are "unimportant" or outliers. Selecting such samples is not very helpful for the classifier to improve the performance.

- **Sparsity analysis**. Eq. (3.14) is derived from sparse representation with only selected samples being considered in the linear combination. As a result, the number of batch size, B_q, is incorporated in our formulation. In other words, the optimal solution is determined with B_q as a hyper-parameter. Rather than our proposed method, Eq. (2.1) is not derived from sparse modeling and therefore the sparsity in their formulation is never analyzed. Besides, they do not incorporate the batch size in their optimization.
- **Efficiency analysis**. In our formulation, we construct matrix Q with only k-nearest neighbors, as illustrated in Eq. (3.6). As a result, this setting makes the quadratic matrix $Q^T Q$ become a sparse matrix, which induces efficient optimization [55]. Moreover, the selective sampling step can also largely reduce the searching space. On the contrary, the k-nearest neighbors strategy cannot be easily adopted in matrix K of Eq. (2.1). This is because this setting will lead to asymmetric property for K. Therefore, the convexity of the formulation will no longer hold, which only results in local minimum solution.

3.2.7. Approximated Solution 2: QP via l_1 Norm Relaxation

We are interested in the sparse solution of Eq. (3.14), which is NP-hard since there is an l_0-norm constraint. If we have $\mathrm{card}(S_U)$ pre-selected unlabeled samples, then we should try $\binom{\mathrm{card}(S_U)}{B_q}$ combinations to select the optimal B_q samples for a query, which is not practical. Therefore, an approximated solution is proposed via l_1-norm relaxation, i.e., we can relax $\|f\|_0$ to $\|f\|_1$. As a result, the problem becomes

$$\begin{aligned} &\hat{f} = \arg\min_f \frac{1}{2} f^T Q^T Q f - f^T (\lambda_1 Q^T + \lambda_2 I) s, \\ &s.t. \|f\|_1 = B_q,\ \mathbf{0} \le f \le \mathbf{1}. \end{aligned} \tag{3.15}$$

It is equivalent to

$$\begin{aligned} &\hat{f} = \arg\min_f \frac{1}{2}\ f^T Q^T Q f - f^T (\lambda_1 Q^T + \lambda_2 I) s, \\ &s.t. \mathbf{1}^T f = B_q,\ Cf \le d, \end{aligned} \tag{3.16}$$

where $C = [-I, I]^T$, $d = [\mathbf{0}, \mathbf{1}]^T$ and I is the identity matrix. Here, we convert the lower and upper bounds of f to be linear inequality constraints. Moreover, $\mathbf{1}^T f = B_q$ is equivalent to

$\|f\|_1 = B_q$ because entries in f are all non-negative values. Hence, the formulation becomes a standard quadratic programming (QP) problem, which can now be solved by the interior-point method [56], [57].

However, l_1-norm cannot guarantee the number of non-zero entries. Since we restrict f to [0, 1], the solution will give us more than B_q non-zero entries. Besides that, since we are only interested in the first B_q samples, we do not want non-selected samples to influence the solution. As a result, we modify the formulation by introducing a parameter λ in the constraint, i.e.,

$$\hat{f} = \arg\min_f \tfrac{1}{2} f^T Q^T Q f - f^T(\lambda_1 Q^T + \lambda_2 I)s,$$
$$s.t., \mathbf{1}^T f = \lambda B_q,\ Cf \leq d, \tag{3.17}$$

where we choose λ between 0 and 1. Decreasing λ will generate less non-zero entries in the solution. We can adjust λ so that the solution only contains B_q non-zero entries. This problem can be solved by using a simple bisection algorithm to iteratively search the proper λ, which is illustrated in Algorithm 2. We name this method as sparse modeling via quadratic programming (SMQP), which uses quadratic programming to solve the sparse modeling problem. Usually, the optimal solution will be achieved in no more than 10 iterations from our simulations.

Algorithm 2: SMQP

Input: original uncertainty score s, similarity matrix Q, labeled set L, pre-selected unlabeled set S_U.
Initialization: Set λ = 0.5, $lb = 0$, $ub = 1$.
Solve $\hat{f}$ by Eq. (3.17).
while $\|\hat{f}\|_0 \neq B_q$ **do**
if $\|\hat{f}\|_0 > B_q$ **do**
$ub = \lambda$.
else
$lb = \lambda$.
end if
$\lambda = (lb + ub)/2$.
Solve $\hat{f}$ by Eq. (3.17).
end while

Sort $\hat{f}$ in descending order and move the first B_q indices of $\hat{f}$ from S_U to L.

Output: updated labeled set L.

3.2.8. Experimental Results and Analysis

1) Datasets

A) Cam-Trawl Fish Dataset

The Cam-Trawl Fish dataset contains 1026 images with 5 classes. The number of samples of each class is shown in Table 3.3, with example images shown in Figure 3.9.

Table 3.3: Data distribution of Cam-Trawl Fish Dataset

Class name	# of samples
Eulachon	119
Pollock	416
Rockfish	216
Salmon	159
Squid	116

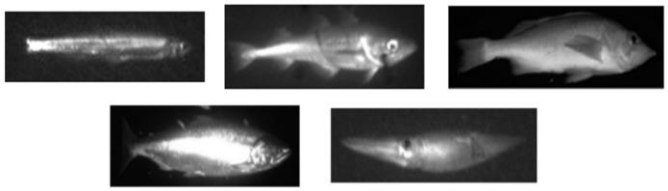


Figure 3.9: Example images of Cam-Trawl Fish dataset.

B) Chute Fish Dataset

The Chute Fish dataset, which is a much larger dataset containing 5032 images with 27 classes. Data distributions are shown in Table 3.4, with example images shown in Figure 3.10.

Table 3.4: Data distribution of Chute Fish Dataset

Species	Size	Species	Size
Arrowtooth Flounder	436	Pacific Halibut	59

Atka Mackerel	307	Pacific Ocean Perch	227
Bathymaster Signatus	97	Pacific Octopus	31
Berryteuthis Magister	131	Paragorgia Arborea	61
Black Spotted Rockfish	139	Prowfish	54
Dover Sole	55	Rex Sole	433
Dusky Rockfish	67	Sablefish	50
Flathead Sole	511	Short Spine Thornyhead	213
Giant Grenadier	40	Shortraker Rockfish	154
Gorgonocephalus Eucemis	31	Strongylocentrotus Sp.	57
Harlequin Rockfish	21	Sturgeon Poacher	24
Northern Rock Sole	753	Walleye Pollock	280
Northern Rockfish	385	Yellow Irish Lord	193
Pacific Cod	223		

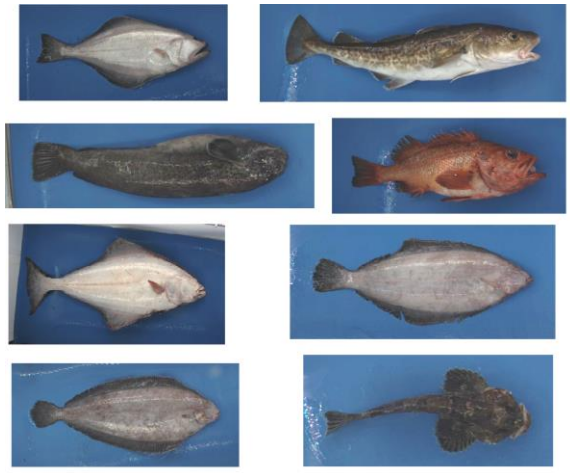


Figure 3.10: Example images of Chute Fish dataset.

2) Experiment Setup

For each dataset, we split the data into 4 parts: seed set, unlabeled set, validation set and testing set, denoted as $\{L, U, V, T\}$. L is the initial seed set, which is used for training the initial classifiers; U is the unlabeled data pool for sample selection; V is used for parameter tuning;

and T is used for evaluating the performance of the re-trained classifiers. For each dataset, we split the data as follows: in each class, 3 samples are used as the seed set; half of the samples are used as unlabeled data; one quarter as the validation set and the remaining are used as the testing set. During each experiment, the data is split randomly.

For each dataset, two different types of feature extraction methods are adopted. One is based on the traditional extraction method and the other is convolutional neural networks (CNNs) based [60]. For the traditional extraction method, we follow the bag-of-features (BoF) framework [58], [59] based on two level codebook learning [30]. For the CNN based feature extraction method, we use the output of pre-logits layer of inception-resnet-v2 [60] as the feature vector. Before extracting CNN features, we use transfer learning on each dataset to achieve better feature representation. The feature space used for the four datasets is summarized in Table 3.5.

There are 8 methods used for evaluation in the experiments: 1) BvSB [34], which chooses the B_q samples based on top highest uncertainty scores; 2) RAND, which randomly selects B_q samples for a query; 3) SMGS, which is the proposed approximated approach using sparse modeling via greedy search; 4) VS [39], which incorporates diversity for a query via version space reduction; 5) USDM [40], which is uncertainty sampling based active learning with diversity maximization; 6) SMQP, which is the proposed approximated approach using sparse modeling via quadratic programming; 7) MMC [47], which is active learning with maximum model change; 8) EER [48], which is active learning with expected error reduction.

For the parameter settings, we fix the cost $C_p = 1$ in the SVM for all experiments. Also, we fix $d_{\text{thresh}} = 0.2$ in the selective sampling step for the sparse modeling. We also set $\lambda_1 = 1$ in the refinement of sparse modeling to be fixed. Other parameters are empirically tuned according to the performance in the validation set V. We tune the standard deviation σ of the similarity matrix Q from {0.25, 0.5, 1, 2}, λ_2 in the refinement of sparse modeling from {0.1,1,5,10} and the number of neighbors $\text{card}(N_j)$ of Q from {5,10,20}.

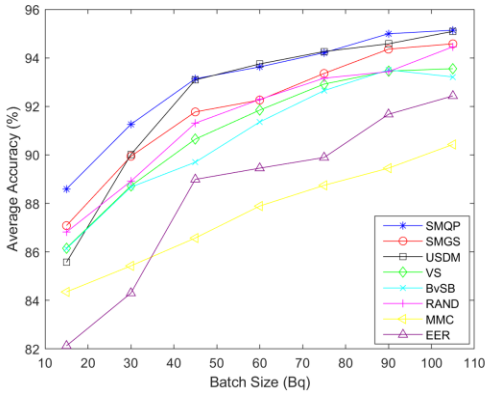
Table 3.5: Feature description

Name	Feature size

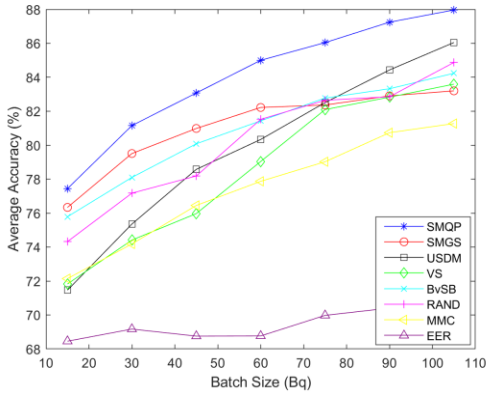
Cam-Trawl Fish (BoF)	7168
Cam-Trawl Fish (CNN)	1536
Chute Fish (BoF)	7168
Chute Fish (CNN)	1536

3) Performance Comparison with Different Batch Sizes

We run the algorithms with different batch sizes from 15 to 105 with 15 increments for each experiment and report the average accuracy for all the 8 methods. We set the seed size $c = 3$ in this experiment. Each result is based on an average of 10 runs of the same setting. Figure 3.11 compares the performance of 8 active learning algorithms for image classification on the datasets. Generally, SMQP and SMGS outperform other methods. To be specific, SMQP gives robust results on different batch size and SMGS also gives promising results. However, since no optimal solution is guaranteed in SMGS, in a few cases the accuracy of SMGS is slightly lower than BvSB method, particularly for Chute Fish dataset when the batch size is over 75. For MMC and EER methods, although they are optimized for sequential sample selection, they are not taking advantage of the unlabeled pool data for batch mode settings. As a result, they perform even worse than RAND. This is because the sequentially selected samples in MMC and EER have much redundant information which gives less information for re-training the classifiers than random selection. As for VS method, it also incorporates diversity in the experiment design and use version space reduction to deal with binary problem. However, the results show that there is no big improvement than BvSB method. A possible reason is that version space reduction may be not suitable for multi-class problems. Generally, the performance of USDM is usually much better with large batch size than with small batch size. This is because USDM always tends to look for isolated distinct samples first, which has been discussed in the previous sections.



(a) Cam-Trawl Fish

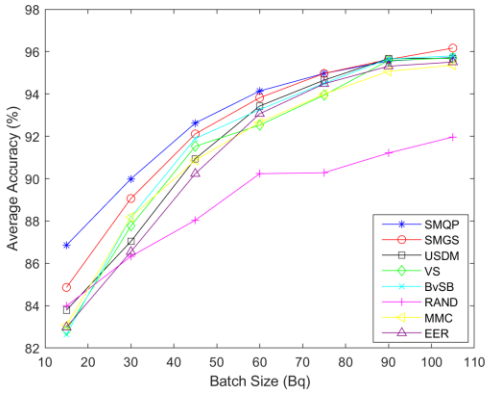


(b) Chute Fish

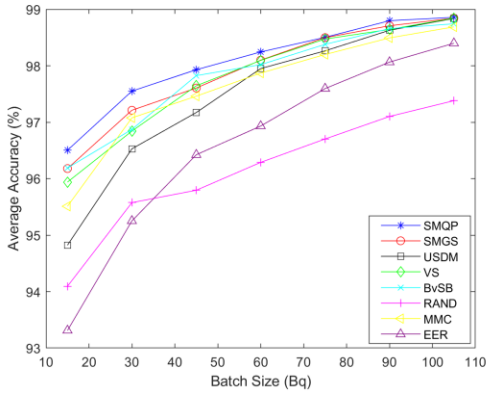
Figure 3.11: Average accuracy with seed size $c = 3$ on two datasets using traditional features.

4) Performance Comparison Using Different Features

In this subsection, we compare the performance of 8 active learning methods related to CNN based features. Figure 3.12 shows the experimental results on the four datasets. More details about CNN feature representation can be found in Table V. Since transfer learning is conducted on the dataset before feature extraction, the classifiers are more robust with the same size of seed set compared to using traditional features in the previous subsection. This experiment demonstrates that when a better feature is used, the performance of an active learning algorithm usually improves. Same as before, SMQP outperforms the other competitors consistently using different features. When the batch size increases, the performances saturate among several different methods with robust classifiers.



(a) Cam-Trawl Fish

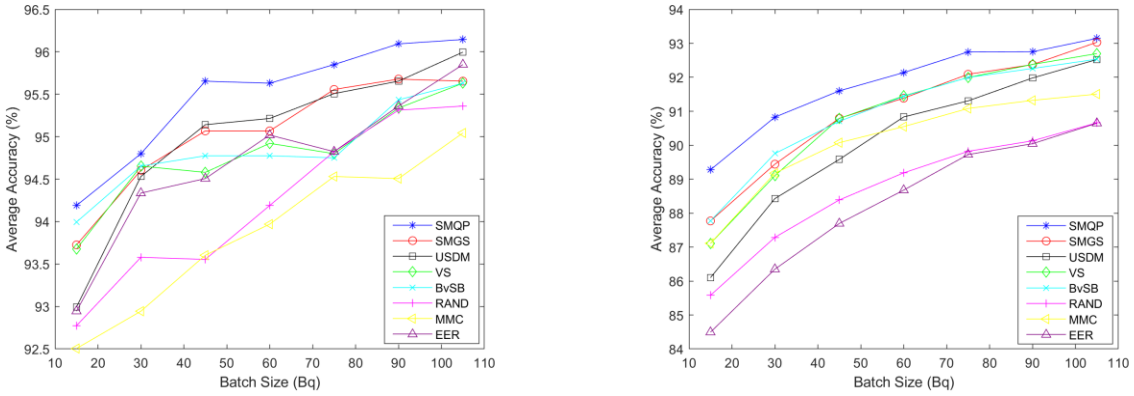


(b) Chute Fish

Figure 3.12: Average accuracy with seed size $c = 3$ on two datasets using CNN features.

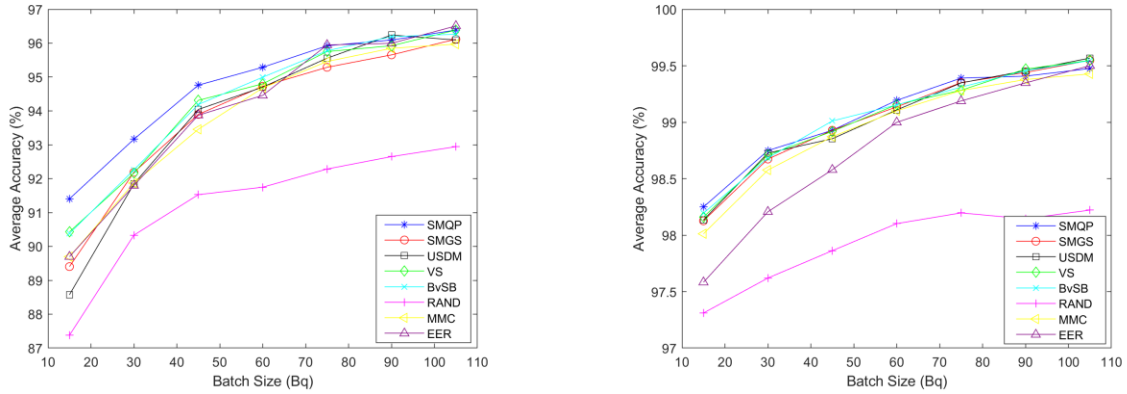
5) Performance Comparison Using Different Seed Sizes

In this subsection, we examine the impact of the seed size by changing $c = 9$ on these two datasets. We keep other settings unchanged and evaluate the performance with the batch sizes varying from 15 to 105. Both traditional features and CNN features are used for complete comparison. The results are shown in Figure 3.13 and Figure 3.14. From the results, we can see that the proposed methods are consistently favorable, which further indicates that leveraging the unlabeled pool data does help improve the active learning performance.



(a) Cam-Trawl Fish (b) Chute Fish

Figure 3.13: Average accuracy with seed size $c = 9$ on two datasets using traditional features.



(a) Cam-Trawl Fish (b) Chute Fish

Figure 3.14: Average accuracy with seed size $c = 9$ on two datasets using CNN features.

6) Significance Test Analysis

Since USDM also incorporates diversity in the minimization, we use paired sign test to verify whether SMQP has a significant improvement over USDM. For each testing sample, the outcome of the two classifiers have 4 possibilities: 1) both USDM and SMQP make correct predictions; 2) USDM makes a correct prediction while SMQP makes a wrong prediction; 3) SMQP makes a correct prediction while USDM makes a wrong prediction; 4) both USDM and SMQP make wrong predictions. We use a 2×2 contingency table to tabulate the outcomes of two classifiers on all testing samples, as follows.

Table 3.6: Contingency table for two classifiers

		SMQP classifier	
		Correct	Wrong
USDM classifier	Correct	a_1	a_2
	Wrong	a_3	a_4

The null hypothesis H_0 is that these two classifiers are the same while the alternate hypothesis H_1 is that SMQP is better than USDM. The p-value is defined as the probability that the same as or more extreme cases than the actual observed results occur, when the null hypothesis is true. A smaller p-value means that SMQP classifier is more likely to be better than USDM classifier. The p-value can be formulated as follows,

$$p = Pr(X \geq a_3) = 1 - Pr(X < a_3)$$
$$= 1 - \sum_{i=0}^{a_3-1} \binom{a_2+a_3}{i} 0.5^i \, (1-0.5)^{a_2+a_3-i}. \quad (3.18)$$

We report all p-values based on the sign test between USDM and SMQP classifiers on two datasets using traditional features in Table 3.7.

For SMQP method, the smaller the p-value is, the more significant the improvement is over USDM method. From the table, we can see that SMQP method has a significant improvement over USDM method especially for small batch size and seed number. With large batch size and seed number, some p-values are relatively large. This is because the classifier becomes more robust with the increase of the batch size and seed number.

Table 3.7: p-Values between SMQP and USDM for four datasets with different batch sizes

Bq	15	30	45	60	75	90	105
Cam-Trawl Fish (c)	7.69E-05	2.69E-04	4.55E-02	8.23E-02	8.68E-02	9.75E-02	5.00E-02
Cam-Trawl Fish (3×c)	8.60E-03	1.10E-03	5.42E-03	5.39E-02	2.02E-02	5.63E-02	3.85E-02
Chute Fish (c)	0	0	0	0	0	0	1.11E-16
Chute Fish (3×c)	0	0	0	0	0	1.87E-08	1.51E-07

Chapter 4 - Length Measurement

4.1. Coarse-to-Fine Segmentation Refinement

The flowchart of the proposed coarse-to-fine refinement method is shown in Figure 4.1. More details of the method will be illustrated in the following subsections.

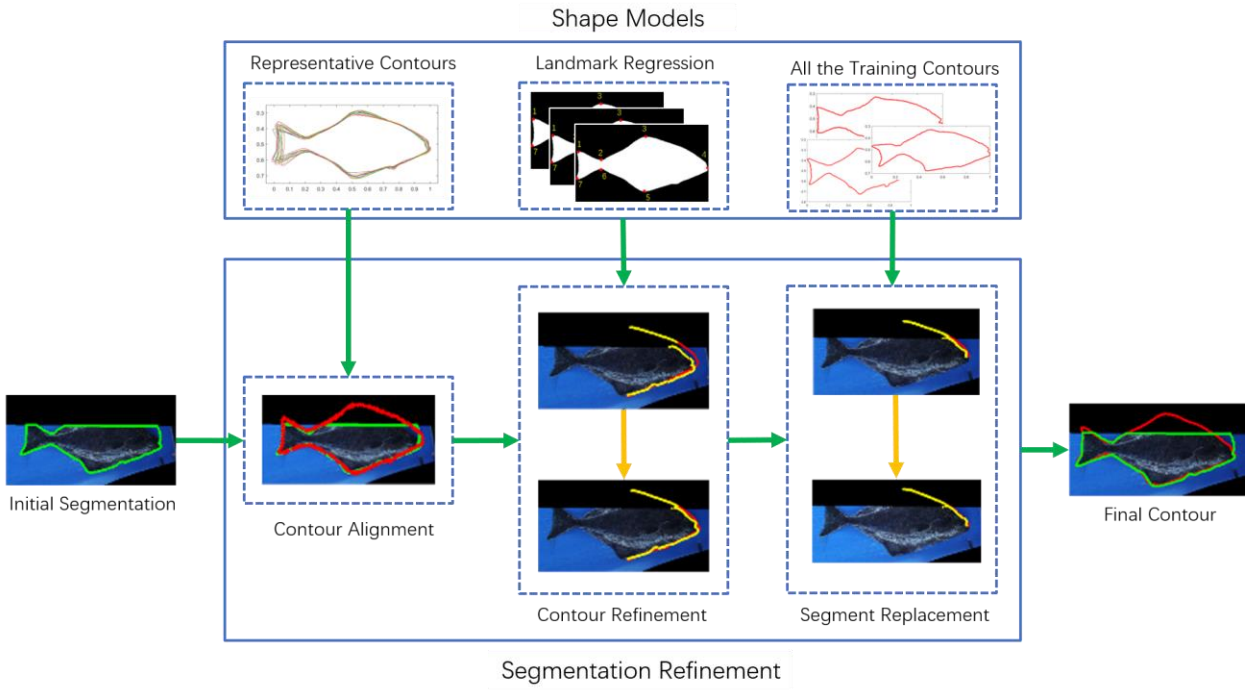


Figure 4.1: The flowchart of the contour refinement.

4.1.1. Shape Models Training

Before processing the segmentation refinement, we illustrate how we train the shape models in this subsection. As shown in the top of Figure 4.1, the shape models contain 3 components, i.e., representative contours, landmark regression and all the training contours. For each training contour, we uniformly sample the contour points to a fixed length size, 1000. Then twenty representative contours are generated using k-means algorithm based on all the training contours, as shown in Figure 4.2.

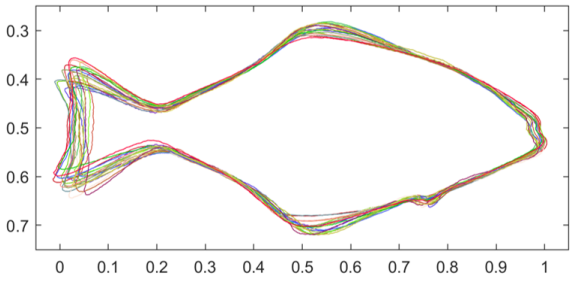


Figure 4.2: Twenty representative contours generated by k-means.

As for landmark regression model, we first define seven landmarks with highest curvatures, on the contour of the fish segmentation. Landmarks 1 and 7 are located at the end of the tail; landmarks 2 and 6 are located at the connection between the fish tail and fish body; landmarks 3 and 5 are the turning points of the fish body while landmark 4 is located at the fish mouth. An example is shown in Figure 4.3.

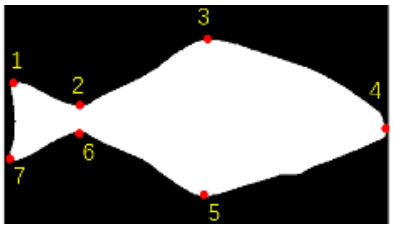


Figure 4.3: An example of 7 landmarks used in the shape model (red dots).

In case of part of the fish is missing, some landmarks may not be accessible in the initial segmentation. To deal with such issues, we train a linear regression to estimate the missing landmark location based on the other 6 landmarks. For example, if we want to estimate the location of the j-th landmark, the regression model can be expressed as

$$\hat{\mathbf{c}}_j(x) = \arg\min_{\mathbf{c}_j(x)} \left\| \mathrm{B}_j \mathbf{c}_j(x) - \mathbf{v}_j(x) \right\|_2^2, \tag{4.1}$$

$$\hat{\mathbf{c}}_j(y) = \arg\min_{\mathbf{c}_j(y)} \left\| \mathrm{B}_j \mathbf{c}_j(y) - \mathbf{v}_j(y) \right\|_2^2, \tag{4.2}$$

where each row of B_j contains the (x, y)-coordinates of all the other 6 landmarks except the j-th landmark; $\mathbf{v}_j(x)$ and $\mathbf{v}_j(y)$ are (x, y)-coordinates of the j-th landmark of all training data, respectively; $\hat{\mathbf{c}}_j(x)$ and $\hat{\mathbf{c}}_j(y)$ are the learned coefficients of the regression model.

In the testing stage, we can estimate the missing location of the j-th landmark by

$$\mathrm{v}_j^*(x) = \mathbf{b}_j^T \hat{\mathbf{c}}_j(x), \tag{4.3}$$

$$\mathrm{v}_j^*(y) = \mathbf{b}_j^T \hat{\mathbf{c}}_j(y), \tag{4.4}$$

where $\mathbf{b}_j^T$ is the other 6 landmark locations except the j-th landmark in the testing data. All the information in this subsection we generated from the training data are denoted as shape models, which will be used in our refinement processes.

4.1.2. Contour Alignment

It is a challenging task to align the initial segmentation mask using shape models since water drops and missing part can largely affect the initial segmentation. Here, we propose an iterative algorithm to align the segmentation mask with generate contour models via a weighted affine transform. Given the pre-trained shape models, the affine transform [52] can be estimated by

$$\hat{\mathrm{H}} = \arg\min_{\mathrm{H}} \sum_i \|\mathrm{H}\mathbf{p}_i - \mathbf{p}_i^m\|_2^2, \tag{4.5}$$

where $\mathbf{p}_i$ is the contour point in the initial segmentation without touching the image boundary and $\mathbf{p}_i^m$ is a contour point that closest to $\mathbf{p}_i$ from the m-th ($m = 1,2,\ldots,20$) average contour model as defined in Section 4.4.1. If we concatenate the parameters in H to a vector $\mathbf{h}$, then the affine transform can be estimated by a least square problem as

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \|\mathrm{A}\mathbf{h} - \mathbf{p}^m\|_2^2, \tag{4.6}$$

where A contains the information of (x, y)-coordinates of all contour points of the initial segmentation (from 1 to N), i.e.,

$$\mathrm{A} = [\mathrm{A}_1; \ldots; \mathrm{A}_i; \ldots; \mathrm{A}_N],$$
$$\mathrm{A}_i = \begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_i & y_i & 1 \end{bmatrix}. \tag{4.7}$$

If water drops occur along the segmentation contour, then the affine transform can be largely affected by these unreliable contour points. To avoid the influence by water drops, we set the gradient magnitude of the raw image as a weight for each contour point from the initial

segmentation since water drop regions always have low gradient magnitudes. Then a weighted affine transform is estimated by

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \|\mathrm{W}(\mathrm{A}\mathbf{h} - \mathbf{p}^m)\|_2^2, \tag{4.8}$$

where W is a diagonal matrix with the diagonal elements equal to the gradient magnitudes of the contour points.

We perform the weighted affine transform estimation iteratively until convergence. To be specific, for the t-th iteration, the closest points from the m-th average contour model to match the previous transformed data $\mathrm{A}\mathbf{h}^{t-1}$ are selected as $(\mathbf{p}^m)^t$. We try all the average contour models and select the best one that provides the smallest distance error between transformed data $\mathrm{A}\mathbf{h}$ and $\mathbf{p}^m$ after convergence. An example of contour alignment is shown in Figure 4.4.

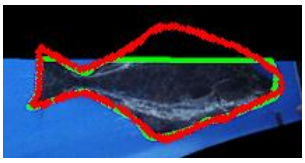


Figure 4.4: Contour alignment. Green points are the contour points from the initial segmentation. Red contour is the best match from the average contour models.

4.1.3. Contour Refinement

For the coarse level of contour alignment, the initial segmentation contour is represented by a general fish shape defined by the chosen average contour model, which cannot precisely match the segmented fish. In this section, the segmentation contour is refined with more local details recovered at two finer levels.

Denote the contour points from the selected average contour model in the previous section as generated contour points (red curve in Figure 4.4). We want to project each generated contour point on the pixel with highest gradient magnitude of the raw image along the normal vector (sky-blue), since we believe the ground truth segmentation boundaries of the raw fish always have high gradient magnitudes. For each generated contour point, we then search, from fish mid-line to slightly outside the red contour, the highest gradient pixel along the normal vector direction as the projected contour point $\tilde{\mathbf{p}}$ which are shown in Figure 4.5 in yellow. If the highest gradient

magnitude is smaller than a threshold σ, it means that the searched projected contour point is unreliable. Then we just use the initial generated contour point as the projected contour point instead.

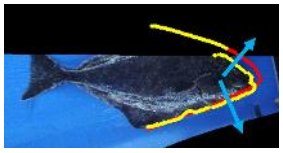


Figure 4.5: An illustration of the search of projected contour points. Each generated contour point (red) is projected on the projected contour point (yellow) along its normal vector (sky-blue). Only two contour segments are shown in the figure for illustration.

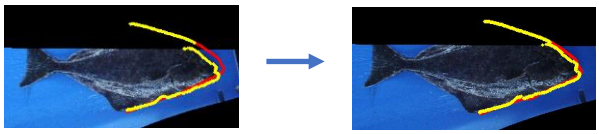


Figure 4.6: Contour refinement. An affine transform is estimated to minimize the distance between generated contour segments (red) and projected contour segments (yellow) for each two adjacent contour segments.

For the finer level refinement, a weighted affine transform is estimated only for two adjacent contour segments as shown in Figure 4.6. To be specific, the affine transform is estimated from

$$\hat{\mathbf{h}}_j = \arg\min_{\mathbf{h}_j} \left\| \mathrm{W}_{j-1}\left(\mathrm{A}_{j-1}\mathbf{h}_j - \tilde{\mathbf{p}}_{j-1}\right)\right\|_2^2 + \left\| \mathrm{W}_j\left(\mathrm{A}_j\mathbf{h}_j - \tilde{\mathbf{p}}_j\right)\right\|_2^2, \tag{4.9}$$

where $\tilde{\mathbf{p}}_j$ are the projected contour points between the j-th landmark and $(j+1)$-th landmark; A_j contains the information of (x, y)-coordinates of generated contour points between the j-th landmark and $(j+1)$-th landmark and W_j is the diagonal matrix with the diagonal elements equal to the gradient magnitudes of projected contour points. We can see that the same affine transform is shared between two adjacent contour segments.

Additionally, if the j-th landmark is missing, the above estimation is not reliable. To deal with such issue, a regularization term is added to the cost function to restrict the missing landmark location to follow the general fish shape from shape models, i.e.,

$$\hat{\mathbf{h}}_j = \arg\min_{\mathbf{h}_j} \left\| \mathrm{W}_{j-1}\left(\mathrm{A}_{j-1}\mathbf{h}_j - \tilde{\mathbf{p}}_{j-1}\right)\right\|_2^2$$

$$+\left\|\mathrm{W}_j\left(\mathrm{A}_j\mathbf{h}_j-\tilde{\mathbf{p}}_j\right)\right\|_2^2+\lambda\left\|\left(\mathrm{D}_j\mathbf{h}_j-\mathbf{v}_j^*\right)\right\|_2^2, \tag{4.10}$$

$$\mathrm{D}_j=\begin{bmatrix} v(x_j) & v(y_j) & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & v(x_j) & v(y_j) & 1 \end{bmatrix}, \tag{4.11}$$

where $\mathbf{v}_j^*$ is the estimated location of the j-th landmark which can be computed by Eq. (4.3) and Eq. (4.4) from the average contour models and D_j contains the information of the j-th landmark coordinates from generated contours. Then for the next iteration, the j-th generated contour segment can be updated with the estimated affine transform,

$$\hat{\mathbf{p}}_j=\mathrm{A}_j\hat{\mathbf{h}}_j. \tag{4.12}$$

At the finest level, we search all the training data to find the best contour segment from shape models to match the projected contour segment, i.e.,

$$\hat{\mathbf{p}}_j^n=\arg\min_{\mathbf{p}_j^n}\left\|\mathbf{p}_j^n-\tilde{\mathbf{p}}_j\right\|_2^2, \tag{4.13}$$

where $\mathbf{p}_j^n$ is the j-th contour segment of the n-th data in the training dataset. Then we replace the generated contour segment $\hat{\mathbf{p}}_j$ with $\hat{\mathbf{p}}_j^n$. An example is shown in Figure 4.7.

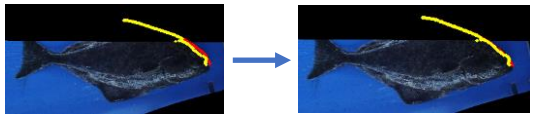


Figure 4.7: Contour segment replacement. Use the best contour segment chosen from all the training data (right red) to replace the generated contour segment (left red) to match the projected contour segment (yellow).

Note that for each iteration, the affine transform $\hat{\mathbf{h}}_j$ is estimated alternatively with $j=1,2,\ldots,7$. Then we alternatively update generated contour points between finer and finest levels iteratively until convergence.

4.1.4. Experimental Results and analysis

1) Implementation Details

We manually label 583 segmentation masks in the experiments. 489 segmentation masks are used to train the shape models while the left 94 images are used for testing. To check the efficiency

of the proposed method, we use two initial segmentation masks as comparison, namely double local thresholding (DLT) [11] and fully convolutional networks (FCN) [13]. In the training stage of the FCN, we use random rotation, flip and crop as the data augmentation. For hype-parameters, we set batch size equal to 2, learning rate equal to 1e-4 with Adam Optimizer and set $\lambda = 10$, $\sigma = 0.15$ in the refinement processes.

2) Water Drops

We use the average intersection over union (IOU) to test the performance of the proposed refined segmentation method. The results of refinement are shown in Table 4.1. We can see the average IOU increases a lot with the refinement for different initial segmentations. Although FCN seems more reliable than DLT, the results after refinement are roughly similar, which means the proposed refinement is robust to the initial segmentations.

Table 4.1: Average IOU before and after refinement

Method	Average IOU (%)
DLT [11]	89.65
FCN [13]	92.35
DLT +Refinement	95.04
FCN+ Refinement	94.92

Assume the segmentation mask is acceptable if the IOU is above a certain threshold. We plot the acceptance rate with the increase of the threshold as shown in Figure 4.8. We can see the refined segmentation result is much better than the initial segmentation.

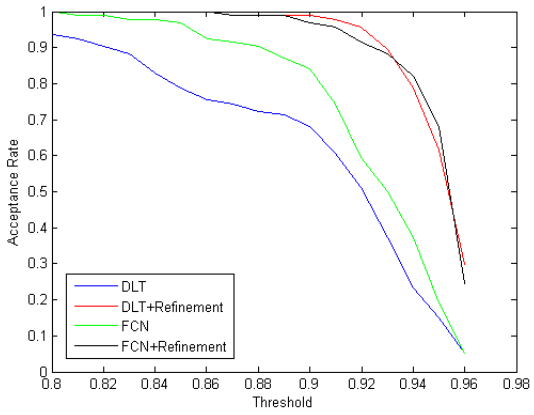


Figure 4.8: Acceptance rate along with the IOU threshold.

We also show some results of the refined segmentation dealing with water drops in Figure 4.10, where the refinement can effectively recover the contour in the blur regions.

3) Missing Part Recovery

To test the performance of the proposed method when a portion of the fish body is out of the camera view, we randomly cut off about 1/3 of the fish body ten times for each testing data. For 940 testing data in total, we estimate the segmentation performance with the proposed method. We measure the mean absolute error of the length and IOU of the segmentation, whose results are shown in Table 4.2. We also show some segmentation examples in Figure 4.10.

Table 4.2: Measurement of IOU and length error

Method	DLT+Refinement	FCN+Refinement
Average IOU (%)	87.81	89.28
Mean length error (%)	4.86	4.07

Similarly, we define the acceptance rate for different thresholds of IOU and length error. We plot the results in Figure 4.9, where we can see no large difference with two initial segmentation approaches, i.e., the proposed refinement is also robust to the initial segmentation dealing with missing part recovery.

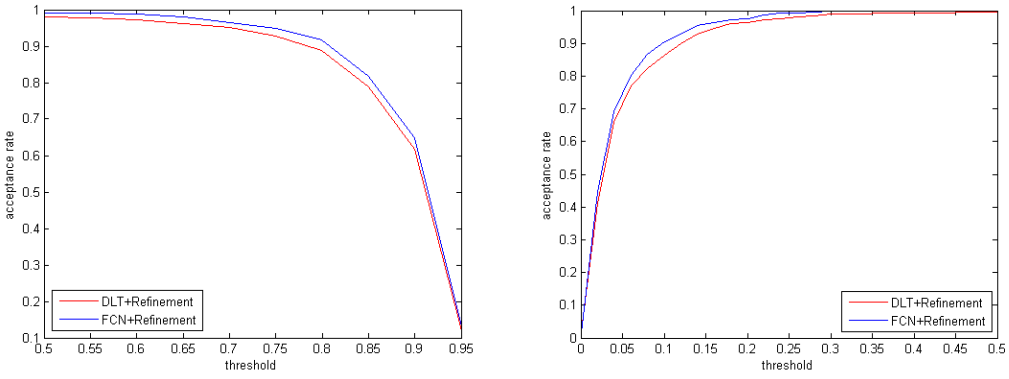
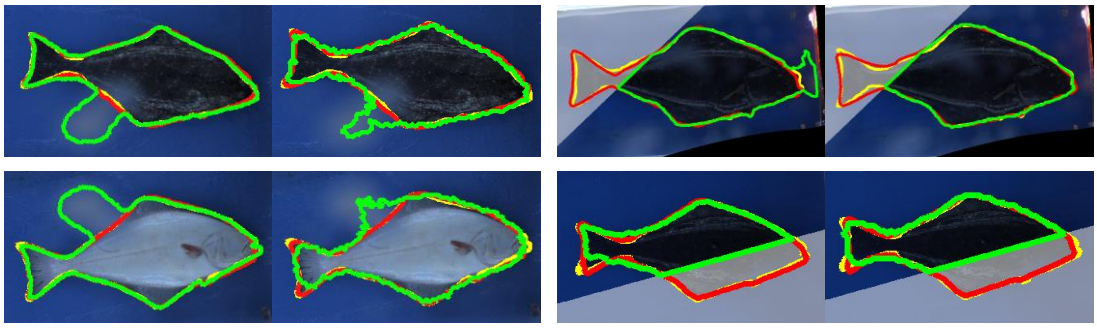


Figure 4.9: Left: acceptance rate along with the IOU threshold. Right: Acceptance rate along with the length error threshold.



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

Figure 4.10: Examples of segmentation refinement. (a) and (b): refinement related to water drops with DLT and FCN, respectively. (c) and (d): refinement related to missing part recovery with DLT and FCN, respectively. Transparent part of the image is the cut-off region to simulate the missing part. Green contours are initial segmentation contours; red contours are refined segmentation contours; yellow contours are ground truth.

Chapter 5 - Underwater Fish Tracking

In this chapter, we design a fish tracking system for underwater environment based on a tracklet graph model. First, the tracklets are generated from successive associated detection results based on intersection-over-union (IoU) and appearance similarity. On one side, due to the unreliable detections and occlusions, the entire trajectory of an object may be divided into several distinct tracklets with a lot of missing detections. On the other side, some detected objects are background objects that are very similar to fish objects, which contributes to high false positive rate. Then, a temporal locality-constrained linear coding (TLLC) is adopted to embed the tracklets into unified feature vectors. A support vector machine (SVM) based classifier is trained to distinguish fish and non-fish tracklets. After remove the non-fish tracklets, a graph model is defined with remaining tracklets. Each tracklet is treated as one vertex in our graph model and the edge between two vertices measures the connectivity of two tracklets. To estimate the connectivity, the TrackletNet is designed for measuring the continuity of two input tracklets, which combines both spatial and temporal information. Given the graph representation, tracking can be regarded as a clustering approach that groups the tracklets into sub-graphs. Each sub-graph can represent a unique tracked fish ID. The framework of our tracking system is shown in Figure 5.1. Specifically, we propose the following contributions.

- To the best of our knowledge, this is the first work to adopt temporal pooling embedding based on bag-of-features (BoF) framework to classify fish and non-fish tracklets to reduce the false positive rate.
- A CNN architecture, called multi-scale TrackletNet, is proposed to measure the connectivity between two tracklets. This network combines trajectory and appearance information into a unified system.
- Our model outperforms some state-of-the-art methods for fish tracking in underwater environment, which provides a novel approach for underwater abundance estimation.

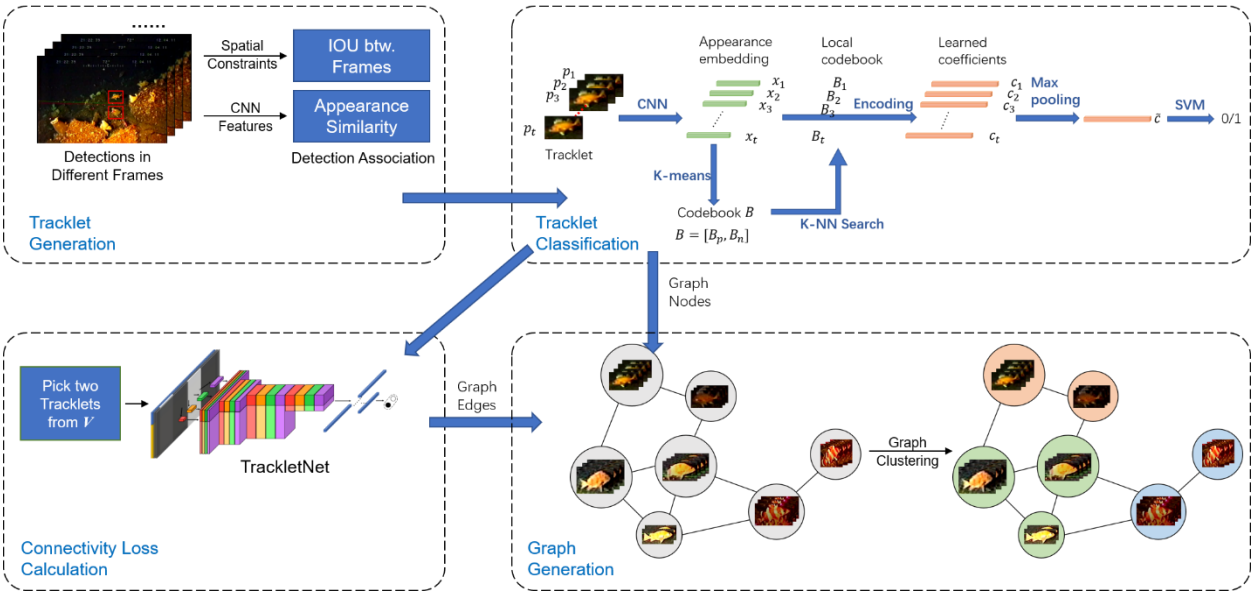


Figure 5.1: The framework of underwater fish tracking.

5.1. Tracklet Classifier

The tracklet classifier contains two components, i.e., tracklet generation and tracklet classification, as shown in Figure 5.2.

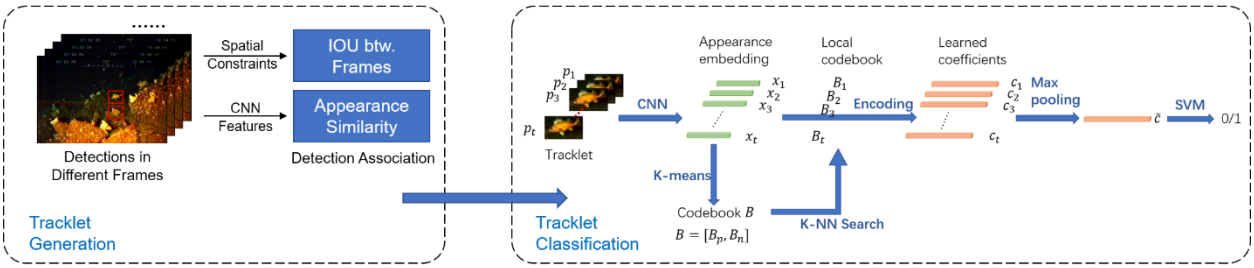


Figure 5.2: The tracklet classifier.

5.1.1. Tracklet Generation

A tracklet contains consecutively detected objects with bounding box information and embedded appearance features. To simplify the generation of tracklets, we associate two consecutive detections based on IOU and appearance similarity in adjacent frames with a high association threshold to ensure the mis-association as small as possible [1], [64].

5.1.2. Appearance Feature Embedding

Appearance feature is an important cue in the object tracking. To make the embedded features discriminative to different objects and similar to the same object, distance metric learning can be used as feature embedding. Followed by FaceNet [109], we learn a CNN embedding with the triplet loss, i.e.,

$$L = \sum_{t}^{|\mathcal{T}|}\left[\left\| x_{t\in\mathcal{T}}^{a} - x_{\tau\in\{\mathcal{T}|\tau\neq t\}}^{p}\right\|_2^2 - \|x_{t\in\mathcal{T}}^{a} - x_{\tau}^{n}\|_2^2 + \alpha\right]_+, \tag{5.1}$$

where $[x]_+ = \max(x, 0)$, $x_{t\in\mathcal{T}}^{a}$ is the embedding of an anchor sample, $x_{\tau\in\{\mathcal{T}|\tau\neq t\}}^{p}$ is the embedding of a positive sample which is the same object as the anchor sample but from other frames, x_{τ}^{n} is an embedding of any negative sample from any frame, and α is the designed margin. Given an input image patch, p, we feed it into the CNN architecture, using the above triplet loss to learn the embedding feature x, i.e.,

$$x = f(p), \tag{5.2}$$

where $f(\cdot)$ is the CNN embedding architecture, which plays a role of metric transformation, and x is the output embedding feature. In the experiment, we use Inception-Resnet-v1 [61] as the CNN architecture.

5.1.3 Temporal Locality-Constrained Linear Coding (TLLC)

After we obtain the appearance features, we want to classify the tracklets based on both spatial and temporal information. Let X_i be a set of appearance features of the tracklet i, which contains the object information in T_i frames, i.e., $X_i = \{x_1, x_2, \dots, x_{T_i}\}$, where each appearance feature $x_t \in \mathbb{R}^d$. Let $l_i \in \{0, 1\}$ be the class label of the tracklet i, which corresponds to non-fish or fish object. Given a testing tracklet j, we want to figure out the label l_j. Usually X_i does not have a fixed size since different tracklets have different time duration T_i, this makes trouble in designing the classifier. One of the simplest strategies is to train a classifier for each x_t, then the majority vote can be adopted for the final prediction. However, the temporal information is not well taken into consideration. Inspired by [108], we propose a temporal locality-constrained linear coding for the tracklet classification (TLLC). First, we can represent the encoding as a feature reconstruction.

Denote $B \in \mathbb{R}^{d \times D}$ as the codebook. Each column in B is regarded as a basis or code with the same dimension as x_t. Then x_t can be reconstructed by using the sparse coding (SC) [111] of B as follows,

$$\hat{C}_i = \min_{C_i} \sum_{t=1}^{T_i} \|Bc_t - x_t\|_2^2 + \lambda \|c_t\|_1, \tag{5.3}$$

where $c_t \in \mathbb{R}^D$ is the sparse coefficient based on the reconstruction error, and $C_i = [c_1, c_2, \dots, c_{T_i}]$. As mentioned in [108], the l_1 norm in the SC is not smooth. Even for very similar feature x, SC might select quite different bases in B. As a result, locality-constrained linear coding (LLC) is adopted, which encourages similar features will have similar codes.

To process LLC, for each input feature x_t, we look for its K nearest neighbors in B, denoted as B_t, where $B_t \in \mathbb{R}^{d \times K}$. B_t is the local codebook corresponding to the input x_t. We also denote the index set of these K nearest neighbors in B as S_{idx}, with $|S_{idx}| = K$. Then the LLC can be represented as

$$\hat{C}_i = \min_{C_i} \sum_{t=1}^{T_i} \|B_t \tilde{c}_t - x_t\|_2^2, s.t. \mathbf{1}^T \tilde{c}_t = 1, \tag{5.4}$$

where $c_t \in \mathbb{R}^K$, and for any $k \in \{1,2, \dots, K\}$, $c_t\big(S_{idx}(k)\big) = \tilde{c}_t(k)$.

After LLC, temporal max pooling is used for the final feature representation, i.e.,

$$\bar{c}_i = \max_{t \in \{1,2,..,T_i\}} c_t, \tag{5.5}$$

To predict the label of the tracklet, SVM is trained based on the input-label pair $(\bar{c}_i, l_i)$.

To make the codebook discriminative for positive and negative samples, B is learned from positive and negative samples separately with $B = [B_p, B_n]$, where B_p represents the codebook learned from positive samples and B_n from negative samples. Both B_p and B_n are learned by the k-means algorithm.

5.2. TrackletNet Tracker (TNT)

After obtaining the tracklets, we propose a multi-object tracking approach, called TrackletNet tracker (TNT) for underwater fish tracking, as shown in Figure 5.3. The TNT contains two

components, i.e., TrackletNet and graph clustering. The details are described in the following section.

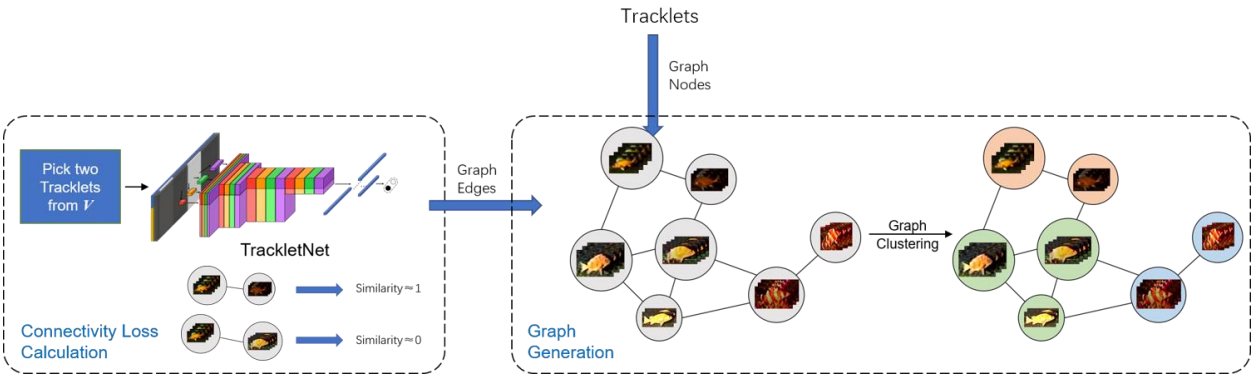


Figure 5.3: The framework of TrackletNet tracker (TNT).

5.2.1. TrackletNet

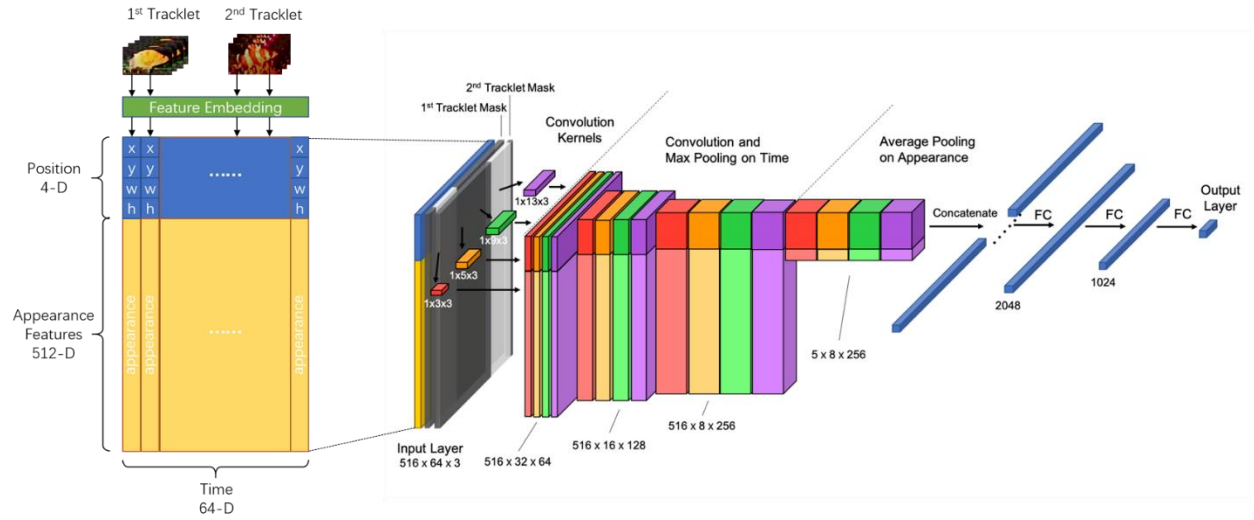


Figure 5.4: Architecture of Multi-scale TrackletNet. First, we extract embedded features from two input tracklets, which include 4D location features and 512D appearance features along the time window of 64 frames. The input tensor has three channels, i.e., one for tracklet embedded features and the other two for binary masks, where white color represents 1 and black color represents 0. Four types of 1D convolution kernels are applied for feature extraction in three convolution layers. For each convolution layer, max pooling is adopted for down-sampling in the time domain. Average pooling is conducted on the dimensions of the appearance feature after Conv3. Then two fully connected layers are conducted to get the final output.

To measure the connectivity between two tracklets, we aggregate different types of information, including temporal and appearance features via the designed multi-scale TrackletNet. The architecture of the proposed TrackletNet is shown in Figure 5.4.

For each frame t, a vector consisting of the bounding box parameters, i.e., (x_t, y_t, w_t, h_t), concatenated by an embedded appearance feature extracted from the FaceNet [109], is used to represent an individual detection from a tracklet. Considering two tracklets with edge-shared in the graph, we concatenate the embedded feature of each detection from these two tracklets inside a time window with a fixed size T. Then the feature space in the time window of the two tracklets is $(4 + d_{ap}) \times T$. As for frames between the two target tracklets, we use a $(4 + d_{ap})$ dimensional interpolated vector instead at each missing frame t. Besides, zero-padding is used for frames after the second tracklet. To better represent the time duration of input tracklets, two binary masks are used as individual channels with $(4 + d_{ap}) \times T$ dimension for each input tracklet. For each frame t, if the detection exists, then we set the t-th column of the binary mask to be all **1** vector; otherwise we set **0** vector instead. As a result, the size of the input tensor of the TrackletNet is $B \times (4 + d_{ap}) \times T \times 3$, where B is the batch size and 3 indicates the number of channels, one for the embedded feature space and the other two for the binary masks.

TrackletNet contains three convolution layers Conv1, Conv2, Conv3, one average pooling layer AvgPool, and two fully connected layers FC1, FC2. For each convolution layer, four different sizes of kernels are used, i.e., 1×3, 1×5, 1×9, 1×13. Note that our convolution is only in the time domain, which can measure the continuity for each dimension of the feature. Different sizes of kernels will look for feature changes in different scales. The large kernels have the ability to measure the continuity of two tracklets even if they are far away in the time domain, while small kernels can focus on appearance difference if input tracklets are in small pieces. Each convolution is followed by one max pooling layer which down-samples by 2 in the time domain. After Conv3, we take the average pooling on appearance feature dimensions. AvgPool plays a role of the weighted majority vote on the discontinuity of all appearance dimensions. Then we concatenate all features and use two fully connected layers for the final output. The output is defined as the similarity between the two input tracklets, which ranges from zero to one.

There are some important properties of the TrackletNet, which are listed as follows.

- TrackletNet focuses on the continuity of the embedded features along the time. Because of the independence among different feature dimensions, no convolution is conducted across the dimensions of the embedded features. In other words, the convolution kernels only capture the dependency along time.
- Binary masks of the input tensor play a role as the tracklet indicator, telling the temporal locations of the tracklets. They help the network learn if the discontinuity of two tracklets is caused by frames without detection or the abrupt changes of the tracklets.
- The network integrates object Re-ID, temporal and spatial dependency as one unified framework.

5.2.2. Graph Definition

Unlike the detection-based graph models, which are computationally expensive and not well utilizing temporal information, our proposed TrackletNet Tracker (TNT), as shown in Figure 5.4, uses tracklets as the vertices in our graph model with edges measuring the similarities between tracklets. From the tracklets, we can infer the objects' moving trajectories for a longer time, and we can also measure how the embedded features of the detections change along the time. Moreover, the number of tracklets is much less than the number of detections, which makes the optimization more efficient. First, we define the notations of the graph in Table 5.1. In the following section, we will discuss in detail about the designed graph model and formulate the tracking as an optimization problem.

Table 5.1: Notations for the tracklet graph model

Symbol	Description
$G(V, E)$	The tracklet graph with the Vertex Set V and the Edge Set E.
e	The edge $e = \{u, w\} \in E$ connected by u, w.
S_i	The cluster i, a subset of V, i.e., $S_i \subseteq V$.
$\pi_{\{u,w\}}$	The edge label with $\pi_{\{u,w\}} = -1$ if $u, w \in S_i$ and $\pi_{\{u,w\}} = +1$ if $u \in S_i, w \in S_j, S_i \cap S_j = \emptyset$.

$S(u)$	The cluster that u belongs to, i.e., $u \in S(u)$.
C_s	The cluster set in the feasible solution, i.e., $C_s = \{S_i \mid i = \{1,2, \dots\}\}$, s.t. $\cup S_i = V, S_i \cap S_j = \emptyset$ if $i \neq j$; $\forall \{u,w\} \in E$, if $\pi_{\{u,w\}} = -1, S(u) = S(w)$; $\forall \{u,w\} \in E$, if $\pi_{\{u,w\}} = +1, S(u) \neq S(w)$.
$S_{bef}(u,t_u)$	A subset of $S(u)$, i.e., $S_{bef}(u,t_u) = \{w \in S(u) \mid t_w \leq t_u\}$, where t_u is the ending frame of the tracklet u.
$S_{aft}(u,t_u)$	A subset of $S(u)$, i.e., $S_{aft}(u,t_u) = \{w \in S(u) \mid t_w > t_u\}$, and $S(u) = S_{bef}(u,t_u) \cup S_{aft}(u,t_u)$.
$N(u)$	A set of tracklets, i.e., the neighboring tracklets that connected with u.

Vertex Set. A finite set V in which every element $u \in V$ represents a tracklet across multiple frames, i.e., a set of consecutive detections of the same object along time. For each detection, we define the bounding box with five parameters, i.e., the center of the bounding box (x_t, y_t), the width and height (w_t, h_t), and the frame index t. Besides the bounding box of the detection, we also extract appearance features [109] for each detected object at frame t. Note that because of unreliable detections, an entire trajectory of an object may be divided into multiple pieces of tracklets.

Edge Set. A finite set E in which every element $e \in E$ represents an edge between a pair of two neighboring tracklets $u, w \in V$ in the time domain, i.e., $min_{t_u \in T(u), t_w \in T(w)} |t_u - t_w| \leq \delta_t$, where $T(u)$ is the set of frame indices of the tracklet u. For tracklets that are far away from each other, the edge is not considered between them since not enough information can be utilized for measuring their relationship.

Then, a connectivity measure p_e is defined to represent the similarity of the two tracklets connected by the edge $e \in E$. The edge cost c_e is defined as

$$c_e = \log\left(\frac{p_e}{1-p_e}\right), \tag{5.6}$$

which represents the cost of cutting the edge. Moreover, the connectivity is defined to be 0 if two tracklets have overlap in the time domain since they must belong to distinct objects. This is because an object cannot appear in two tracklets at the same time. Note that p_e is measured by our designed TrackletNet.

5.2.3. Tracklet Clustering

After the tracklet graph is built, we acquire the object trajectories by clustering the graph into different sub-graphs. Given a tracklet graph $G(V, E)$, for every edge $e \in E$, a cost c_e is to be paid based on the predicted clustering labels. Let u and w be arbitrary neighboring vertices connected by the edge e. Let $\pi_e = -1$ if u and w are clustered in the same track and $\pi_e = +1$ if they are clustered in distinct tracks. Then the cost will be paid if the similarity $p_e > 0.5$ and $\pi_e = +1$, or the similarity $p_e < 0.5$ and $\pi_e = -1$. Therefore, the clustering cost on the edge e can be defined as $\pi_e \cdot c_e$. As a result, the objective function can be defined to minimize the total clustering cost L_G on all graph edges as follows,

$$\min_{\pi_e \in \{+1,-1\}} L_G = \sum_{e \in E} \pi_e \cdot c_e, \tag{5.7}$$

subject to

$$\max(\pi_e, 0) \leq \sum_{e' \in C \setminus \{e\}} \max(\pi_{e'}, 0),$$

$$\forall C \in \mathrm{CYCLES}(G), \forall e \in C. \tag{5.8}$$

Here, $\mathrm{CYCLES}(G)$ returns all cycles in a graph. The solution should partition the graph into different clusters.

Note that the costs c_e can be both positive and negative. For tracklets $u, w \in V$ connected by an edge $e = \{u, w\}$, the assignment $\pi_e = -1$ indicates that u and w belong to the same track. Thus, the constraint in the objective function can be understood as follows: If, for any neighboring vertices u and w, there exists a path in G from u to w along which all edges are labeled -1 (indicating that u and w belong to the same track), then the edge $e = \{u, w\}$ cannot be labeled $+1$ (which would indicate the opposite), where the $\max(\cdot)$ function in the equation is to convert the label from -1 to 0. In fact, the constraints from Eq. (5.8) are generalized

transitivity constraints which guarantee that a feasible solution π well defines a decomposition of the graph G into tracks.

Based on the objective function, the graph partition can be formulated as a clustering problem. However, the minimum cost of graph cut problem defined by Eq. (5.7) and Eq. (5.8) is APX-hard [112]. Besides, the number of clusters is unknown in advance. Inspired by [113], [114], we come up with a bottom-up graph clustering approach that uses five operations to find the optimal cut in the local search. The operations are designed to update the given clusters to decrease the total cost for the target tracklet u. To be specific, the five clustering operations are defined as follows.

Assign $O_A\big(u, S(u), S(w)\big)$:

$$\begin{aligned} S(u) &= S(u)\backslash\{u\}, \\ S(w) &= S(w) \cup \{u\}, \\ return\ &S(u), S(w). \end{aligned} \tag{5.9}$$

Merge $O_M\big(S(u), S(w)\big)$:

$$\begin{aligned} S(u) &= S(u) \cup S(w), \\ S(u) &= \{u\}, \\ return\ &S(u). \end{aligned} \tag{5.10}$$

Split $O_{SP}\big(u, S(u)\big)$:

$$\begin{aligned} S_{new} &= S(u)\backslash\{u\}, \\ S(u) &= \{u\}, \\ return\ &S(u), S_{new}. \end{aligned} \tag{5.11}$$

Switch $O_{SW}\big(u, S(u), S(w)\big)$:

$$\begin{aligned} S_1 &= S_{bef}(u, t_u), \\ S_2 &= S_{aft}(u, t_u), \\ S_3 &= S_{bef}(w, t_u), \\ S_4 &= S_{aft}(w, t_u), \\ S(u) &= S_1 \cup S_4, \end{aligned}$$

$$S(w) = S_2 \cup S_3,$$
$$delete\ S_1, S_2, S_3, S_4,$$
$$return\ S(u), S(w). \tag{5.12}$$

Break $O_B(u, S(u))$:

$$S_{new} = S(u) \backslash S_{bef}(u, t_u),$$
$$S(u) = S_{aft}(u, t_u),$$
$$return\ S(u), S_{new}. \tag{5.13}$$

For each operation, the loss change is calculated and the operation that makes the loss decreasing the most will be selected and if only if the loss is non-increasing. The clustering converges if the total loss cannot be further reduced.

5.3. Experimental Results and Analysis

5.3.1. Datasets and Implementation Details

We follow the FaceNet [109] architecture to train the fish appearance embedding. Inception-Resnet-v1 [60] is adopted as the base network and triplet loss is used for distance metric learning. As for the dataset, there are 4 different underwater video sequences. To extract tracklets from the video, a fish detector with YOLOv3 [110] architecture is used to localize the fish. Besides that, the inter-frame association [65] is conducted to extract the object tracklets. In total, there are 1,659 tracklets with 787 positive tracklets and 872 negative tracklets. 17,172 images are contained in the tracklets. To have a better sense of the data distribution, we project the appearance feature of each image onto the 2-dimensional space by PCA for visualization, as shown in Figure 5.5. We can see that the fish and non-fish data are distributed in different clusters. A simple linear classifier may be not good enough for the classification.

In the tracklet classification with TLLC, some default parameters are set as follows. The size of the codebook $D = 1024$; the number of nearest neighbors K in generating the local codebook is set as 5.

5.3.2. Tracklet Classification

We conduct experiments with 3-fold cross validation using different baseline and state-of-the-art methods in the evaluation, as shown in Table 5.2. Two proposed methods are evaluated here, i.e., TLLC and TLLC with class-dependent codebook (CDC). The difference between these two methods is the way of codebook learning. The codebook from TLLC is learned from both positive and negative samples together, while TLLC with CDC learns positive codebook and negative codebook separately. Except the baseline methods, majority and weighted majority vote, a reconstruction based state-of-the-art method is also used for comparison [115]. From the results, we can see that TLLC with CDC gives the best performance. F1 score is used as the performance metric. This is because the number of true negative tracklets is not in our focus in the tracking system.

We also conduct some ablation studies related to codebook size and the number of neighbors in the local codebook. The results are shown in Table 5.3 and Table 5.4. From Table 5.3, we can see that the F1 score increases with larger size of the codebook for both TLLC (without CDC) and TLLC (with CDC). From Table 5.4, we try three different numbers of the neighbors, i.e., 3, 5, 10. The best results are achieved when 5 neighbors are used in generating the local codebook. If the number of neighbors is too many in the local codebook, then some faraway codes also have influence on the input feature x_t, which makes the data sensitive to faraway codes and performance degrades. On the other hand, too less neighbors will cause more reconstruction error.

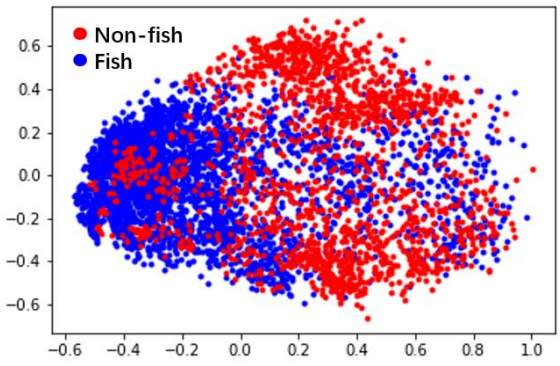


Figure 5.5: The visualization of the data distribution.

Table 5.2: F1 score compared with other methods

Method	F1 (%)
Majority vote	86.4
Weighted majority vote	87.1
Reconstruction based [115]	91.4
TLLC (without CDC)	89.6
TLLC (with CDC)	92.3

Table 5.3: The F1 score of the proposed methods using with different codebook sizes

Codebook Size	TLLC (without CDC)	TLLC (with CDC)
128	87.0	89.2
256	87.5	89.8
512	88.3	89.5
1024	89.6	92.3
2048	91.4	92.5

Table 5.4: The F1 score of the proposed methods using different number of neighbors in the local codebook construction

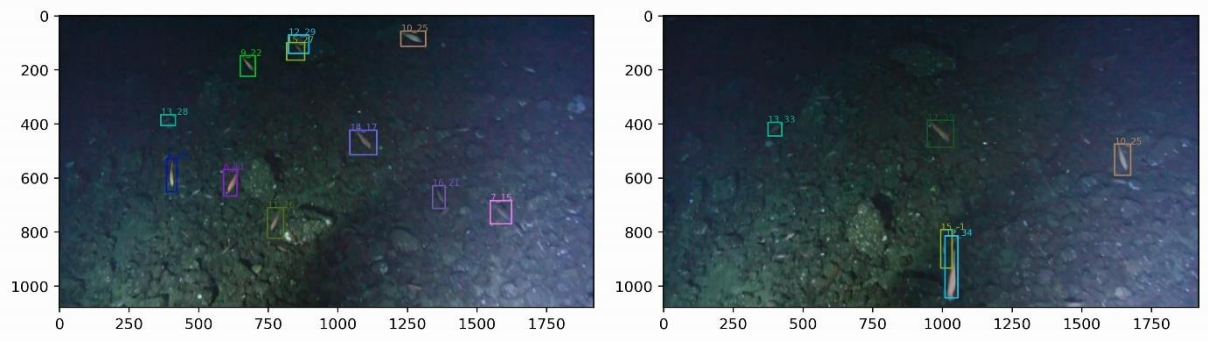
# Neighbors	TLLC	TLLC + CDC
3	88.8	91.1
5	89.6	92.3
10	89.3	91.8

5.3.3. Tracking Performance

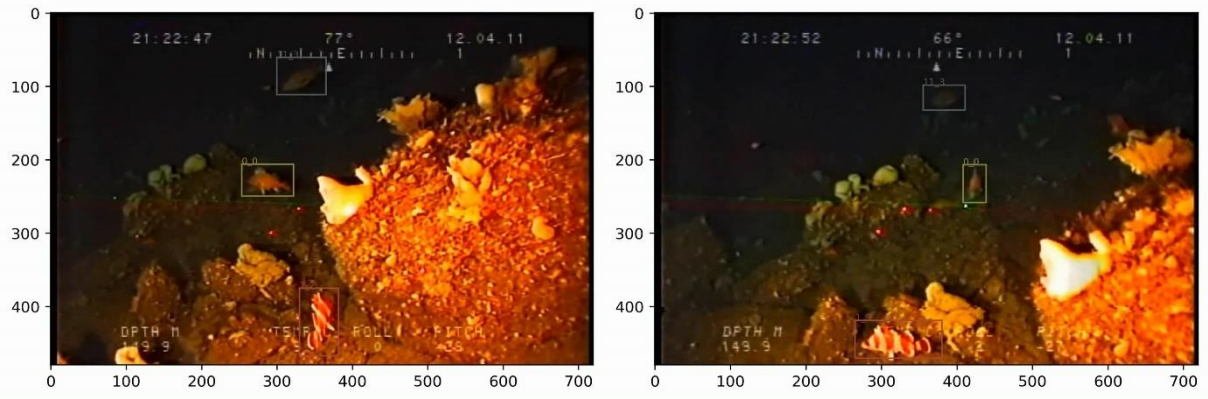
1) Qualitative Results

In Figure 5.6, we show some examples of qualitative results of underwater fish tracking. The first number shown on the top of each bounding box is the fish ID, and the second number represents the tracklet ID. In Figure 5.6-(a), for the targets 12 and 15, we can see although

occlusion happens, the fish can be tracked correctly. For Figure 5.6-(b), we can see the fish can still be tracked even if there is a large change on the pose.



(a)



(b)

Figure 5.6: Examples of qualitative results of fish tracking. (a) Occlusion handling. (b) Pose change.

2) Quantitative Results

We also show some quantitative results of the proposed underwater fish tracking method in Table 5.5. The top part of the table shows the performance for each individual video sequence and the bottom part shows the performance of the all the sequences compared with other state-of-the-art methods. In terms of IDF1 [120] and MOTA [121], our method achieves the best performance.

Table 5.5: Tracking performance compared with other state-of-the-art trackers

Sequence	IDF1	IDP	IDR	MOTA	MOTP	FP	FN	IDs	FM
Fish-01 w/o tracklet classification	93.8	96.1	91.6	89.4	82.9	105	270	1	2
Fish-01 w/ tracklet classification	95.2	96.1	94.4	92.5	82.8	101	165	0	2
Fish-02 w/o tracklet classification	83.2	86.4	80.2	90.1	79.6	118	838	27	27

Fish-02 w/ tracklet classification	86.2	88.2	84.3	91.4	79.9	199	640	22	31
Fish-03 w/o tracklet classification	97.6	99.7	95.6	95.3	79.9	20	278	0	4
Fish-03 w/ tracklet classification	98.9	99.4	98.5	97.9	80.2	36	98	0	5
Fish-04 w/o tracklet classification	89.8	92.6	87.1	93.5	82.0	8	182	2	5
Fish-04 w/ tracklet classification	89.5	94.8	84.8	88.9	82.2	7	318	3	6
Total TNT	91.6	93.5	89.9	93.0	80.7	343	1221	25	44
Total GOG [116]	45.6	51.3	41.0	74.3	78.8	227	4809	817	831
Total IHTLS [117]	71.6	78.4	65.8	71.1	81.4	1403	5061	124	589
Total RMOT [118]	63.7	61.4	66.3	66.6	80.8	4659	2821	144	55
Total DeepSort [119]	72.7	76.5	69.4	87.5	79.9	246	2367	243	269

Chapter 6 – Conclusion and Future Work

6.1. Conclusion

In this dissertation, an automatic video analysis system is proposed for camera-based fisheries surveys. There are three topics in underwater video analysis being investigated, including species identification, segmentation refinement and underwater fish tracking.

To address the challenge of fine-grained species identification task, a novel cut-off augmentation is proposed for the classification when only a small amount of labeled data is available. To efficiently update the classifier given more and more unlabeled data each year, a efficient approach of active learning approach is presented to combine the diversity, density and uncertainty in the sample selection for human labeling. From the experiments, the proposed method achieves very promising result.

To measure the length and size of the fish for on board chute images, a coarse-to-fine segmentation refinement method is designed to get the reliable segmentation contours of the fish even if there is waterdrops on the image or part of the fish is out of the view of the camera. The segmentation refinement method can be built on any kinds of initial segmentation approaches, like double local thresholding or fully convolutional networks. Experiments show the promising results after refinement compared with the initial segmentations.

To estimate the fish abundance, an underwater fish tracking system is designed to address false positive, false negative and ID switches in the complicated underwater environment. First, based on the detection results for each frame, tracklets are generated given the appearance similarity and IOU in adjacent frames. The a tracklet classifier is used to remove false positive tracklets for further analysis. A TrackletNet is designed for measure the similarity between each pair of the tracklets. Afterward, a tracklet graph model is constructed and graph clustering is conducted for partitioning the graph into small groups which represent the final fish ID. Quantitative and qualitative results are shown in the experiments for both tracklet classification and the tracking performance. Compared with other state-of-the-art methods, our proposed method

shows the effectiveness of underwater fish tracking, including handling occlusion and dealing with pose change.

6.2 Future Work

One extension to this work in the future is the capability of combining the species classifier with the segmentation refinement algorithm. Currently, the segmentation refinement algorithm is designed for flat fish, like flounder and halibut. To measure the length and size of other species, a classifier is necessary ahead of the segmentation refinement.

Another track of the future work is to improve the underwater fish tracking algorithm. A more powerful feature extraction method will lead to successful tracking. Currently, the feature embedding is based on the detection bounding box which includes a large region of background irrelevant to the fish body features. Hence, a more advanced feature extraction technique with the help of segmentation to remove background regions is one potential future work.

References

[1] G. Wang, J. N. Hwang, K. Williams, and G. Cutter, “Closed-Loop Tracking-by-Detection for ROV-Based Multiple Fish Tracking,” In Computer Vision for Analysis of Underwater Imagery (CVAUI), 2016 ICPR 2nd Workshop on (pp. 7-12). IEEE, 2016.

[2] G. Wang, J. N. Hwang, C. Rose, and F. Wallace, “Uncertainty sampling based active learning with diversity constraint by sparse selection,” In Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on (pp. 1-6). IEEE, 2017.

[3] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem, “Robust visual tracking via consistent low-rank sparse learning,” International Journal of Computer Vision, vol. 111, no. 2, pp. 171–190, 2015.

[4] S. Avidan, “Ensemble tracking,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 2, pp. 261–271, 2007.

[5] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, “Randomized ensemble tracking,” in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2040–2047.

[6] J. Zhang, S. Ma, and S. Sclaroff, “Meem: Robust tracking via multiple experts using entropy minimization,” in Computer Vision–ECCV 2014. Springer, 2014, pp. 188–203.

[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 37, no. 3, pp. 583–596, 2015.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 9, pp. 1627–1645, 2010.

[9] M.-C. Chuang, J.-N. Hwang, J.-H. Ye, S.-C. Huang, and K. Williams, “Underwater fish tracking for moving cameras based on deformable multiple kernels,” 2016.

[10] Z. Zivkovic, and F. van der Heijden, “Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction,” Pattern Recognition Letters, vol. 27, no. 7, pp. 773- 780, 2006.

[11] M.-C. Chuang, J.-N. Hwang, K. Williams, and R. Towler, “Automatic Fish Segmentation via Double Local Thresholding for Trawl-based Underwater Camera Systems,” IEEE International Conference on Image Processing, Sept. 2011.

[12] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431-3440. 2015.

[13] E. Shelhamer, J. Long, and T. Darrell. "Fully convolutional networks for semantic segmentation." IEEE transactions on pattern analysis and machine intelligence 39, no. 4, pp. 640-651, 2017.

[14] K.-H. Lee, J.-N. Hwang, G. Okapal, and J. Pitton, “Driving recorder based on-road pedestrian tracking using visual slam and constrained multiple-kernel,” in Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014, pp. 2629–2635.

[15] C. Spampinato, Y.-H. Chen-Burger, G. Nadarajan, and R. B. Fisher, “Detecting, tracking and counting fish in low quality unconstrained underwater videos.” VISAPP (2), vol. 2008, pp. 514–519, 2008.

[16] G. Nadarajan, Y.-H. Chen-Burger, R. B. Fisher, and C. Spampinato, “A flexible system for automated composition of intelligent video analysis,” in Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium on. IEEE, 2011, pp. 259–264.

[17] J. Zhou and C. M. Clark, “Autonomous fish tracking by rov using monocular camera,” in Computer and Robot Vision, 2006. The 3rd Canadian Conference on. IEEE, 2006, pp. 68–68.

[18] M. Ravanbakhsh, M. R. Shortis, F. Shafait, A. Mian, E. S. Harvey, and J. W. Seager, “Automated fish detection in underwater images using shape-based level sets,” The Photogrammetric Record, vol. 30, no. 149, pp. 46–62, 2015.

[19] Cohn, David, Les Atlas, and Richard Ladner. "Improving generalization with active learning." Machine learning 15.2 (1994): 201-221.

[20] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active learning with statistical models,” Journal of artificial intelligence research (1996).

[21] L. Zhang, C. Chen, J. Bu, D. Cai, X. He, and Thomas S. Huang, “Active learning based on locally linear reconstruction,” IEEE Transactions on Pattern Analysis and Machine Intelligence 33, no. 10 (2011): 2026-2038.

[22] H. Zhang, H. BvSB, M. Kong, H. Fang, and Z. Zhao, “Active Learning with Sparse Reconstruction.”

[23] Y. Hu, D. Zhang, Z. Jin, D. Cai, and X. He, “Active learning via neighborhood reconstruction,” In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pp. 1415-1421. AAAI Press, 2013.

[24] K. Yu, J. Bi, and V. Tresp, “Active learning via transductive experimental design,” In Proceedings of the 23rd international conference on Machine learning, pp. 1081-1088. ACM, 2006.

[25] A. Gadde, A. Anis, and A. Ortega, “Active semi-supervised learning using sampling theory for graph signals,” Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.

[26] B. Settles, “Active learning,” Synthesis Lectures on Artificial Intelligence and Machine Learning 6.1: 1-114, 2012.

[27] J.-N. Hwang, J. J. Choi, S. Oh, and R. J. Marks, “Query-based learning applied to partially trained multilayer perceptrons,” IEEE Trans. Neural Netw., vol. 2, no. 1, pp. 131–136, 1991.

[28] W.-C. Hu, C.-H. Chen, T.-Y. Chen, D.-Y. Huang, and Z.-C. Wu, “Moving object detection and tracking from video captured by moving camera,” Journal of Visual Communication and Image Representation, vol. 30, pp. 164–180, 2015.

[29] C. Campbell, N. Cristianini, A. Smola, and others, “Query learning with large margin classifiers,” in ICML, 2000, pp. 111–118.

[30] G. Schohn and D. Cohn, “Less is more: Active learning with support vector machines,” in ICML, 2000, pp. 839–846.

[31] S. C. H. Hoi, R.Jin, J. Zhu, and M. R. Lyu, “Semisupervised SVM batch mode active learning with applications to image retrieval,” ACM Transactions on Information Systems (TOIS) 27.3: 16, 2009.

[32] E. Pasolli, F. Melgani, D. Tuia, F. Pacifici, and W. J. Emery, “SVM active learning approach for image classification using spatial information,” IEEE Transactions on Geoscience and Remote Sensing 52.4: 2217-2233, 2014.

[33] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, “Multi-class active learning for image classification,” in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 2009, pp. 2372–2379.

[34] A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos, “Scalable active learning for multiclass image classification,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 11, pp. 2259–2273, 2012.

[35] H. Schütze, E. Velipasaoglu, and J. O. Pedersen, “Performance thresholding in practical text classification,” In Proceedings of the 15th ACM international conference on Information and knowledge management, pp. 662-671. ACM, 2006.

[36] K. Tomanek, and U. Hahn, “A comparison of models for cost-sensitive active learning,” In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pp. 1247-1255. Association for Computational Linguistics, 2010.

[37] B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos, “Active learning for biomedical citation screening,” In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 173-182. ACM, 2010.

[38] Y. Leng, X. Xu, and G. Qi, “Combining active learning and semisupervised learning to construct SVM classifier,” Knowl.-Based Syst., vol. 44, pp. 121–131, 2013.

[39] K. Brinker, “Incorporating diversity in active learning with support vector machines,” in ICML, 2003, vol. 3, pp. 59–66.

[40] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, “Multi-class active learning by uncertainty sampling with diversity maximization,” Int. J. Comput. Vis., vol. 113, no. 2, pp. 113–127, 2015.

[41] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sasrty, “A convex optimization framework for active learning,” In Computer Vision (ICCV), 2013 IEEE International Conference on pp. 209-216, 2013.

[42] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. "SLIC superpixels compared to state-of-the-art superpixel methods." IEEE transactions on pattern analysis and machine intelligence 34, no. 11 pp. 2274-2282, 2012.

[43] J. Shi, and J. Malik. "Normalized cuts and image segmentation." IEEE Transactions on pattern analysis and machine intelligence 22, no. 8 pp. 888-905, 2000.

[44] Y. Y. Boykov, and M-P. Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol. 1, pp. 105-112. IEEE, 2001.

[45] W. Tao, H. J, and Y. Zhang. "Color image segmentation based on mean shift and normalized cuts." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37, no. 5 pp. 1382-1389, 2007.

[46] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour detection and hierarchical image segmentation." IEEE transactions on pattern analysis and machine intelligence 33, no. 5 pp. 898-916, 2011.

[47] W. Cai, Y. Zhang, S. Zhou, W. Wang, C. Ding, and X. Gu, “Active learning for support vector machines with maximum model change,” In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 211-226). Springer, Berlin, Heidelberg, 2014.

[48] B. Yang, J. T. Sun, T. Wang, and Z. Chen, “Effective multi-label active learning for text classification,” In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 917-926). ACM, 2009.

[49] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." arXiv preprint arXiv:1606.00915, 2016.

[50] V. Badrinarayanan, A. Kendall, and R. Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv preprint arXiv:1511.00561, 2015.

[51] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. "Semantic image segmentation via deep parsing network." In Proceedings of the IEEE International Conference on Computer Vision, pp. 1377-1385. 2015.

[52] R. Hartley and A. Zisserman, Multiple view geometry in computer vision. Cambridge university press, 2003.

[53] C. Cortes, and V. Vapnik, “Support-vector networks,” Machine learning 20, no. 3: 273-297, 1995.

[54] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” In Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on, pp. 40-44. IEEE, 1993.

[55] C. C. Paige, and M. A. Saunders, “LSQR: An algorithm for sparse linear equations and sparse least squares,” ACM transactions on mathematical software, 8(1), 43-71, 1982.

[56] A. Altman, and J. Gondzio, “Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization,” Optimization Methods and Software11.1-4: 275-302, 1999.

[57] R. J. Vanderbei, and T. J. Carpenter, “Symmetric indefinite systems for interior point methods,” Mathematical Programming 58.1-3: 1-32, 1993.

[58] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in Workshop on statistical learning in computer vision, ECCV, 2004, vol. 1, pp. 1–2.

[59] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” In Computer vision and pattern recognition, 2006 IEEE computer society conference on, vol. 2, pp. 2169-2178. IEEE, 2006.

[60] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” In AAAI (Vol. 4, p. 12), 2017.

[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, ... and A. Rabinovich, “Going deeper with convolutions,” In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9), 2015.

[62] K. He, X. Zhang, S. Ren, & J. Sun, “Deep residual learning for image recognition,” In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778), 2016.

[63] G. Wang, J.N. Hwang, K. Williams, F. Wallace, and C. S. Rose, “Shrinking encoding with two-level codebook learning for fine-grained fish recognition,” In 2016 ICPR 2nd Workshop on Computer Vision for Analysis of Underwater Imagery (CVAUI) (pp. 31-36), IEEE, 2016.

[64] Z. Zhang, J. Wu, X. Zhang, and C. Zhang, “Multi-target, multi-camera tracking by hierarchical clustering: recent progress on DukeMTMC Project,” arXiv preprint arXiv:1712.09531, 2017.

[65] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J. N. Hwang, “Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features,” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 108-115), 2018.

[66] G. Wang, Y. Wang, H. Zhang, R. Gu, and J. N. Hwang, “Exploit the Connectivity: Multi-Object Tracking with TrackletNet,” arXiv preprint arXiv:1811.07258, 2018.

[67] M. K. Alsmadi, K. B. Omar, S. A. Noah, and I. Almarashdeh, “Fish recognition based on robust features extraction from size and shape measurements using neural network,” Journal of Computer Science, vol. 6, no. 10, p. 1088, 2010.

[68] M.-C. Chuang, J.-N. Hwang, and K. Williams, “Supervised and unsupervised feature extraction methods for underwater fish species recognition,” in Computer Vision for Analysis of Underwater Imagery (CVAUI), 2014 ICPR Workshop on. IEEE, 2014, pp. 33–40.

[69] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1. IEEE, 2005, pp. 886–893.

[70] D.-J. Lee, R. B. Schoenberger, D. Shiozawa, X. Xu, and P. Zhan, “Contour matching for a fish recognition and migration-monitoring system,” in Optics East. International Society for Optics and Photonics, 2004, pp. 37–48.

[71] S. Yang, L. Bo, J. Wang, and L. G. Shapiro, “Unsupervised template learning for fine-grained object recognition,” in Advances in Neural Information Processing Systems, 2012, pp. 3122–3130.

[72] E. Gavves, B. Fernando, C. G. Snoek, A.W. Smeulders, and T. Tuytelaars, “Fine-grained categorization by alignments,” in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1713–1720.

[73] N. Zhang, R. Farrell, F. Iandola, and T. Darrell, “Deformable part descriptors for fine-grained recognition and attribute prediction,” in 2013 IEEE International Conference on Computer Vision. IEEE, 2013, pp. 729–736.

[74] M.-C. Chuang, J.-N. Hwang, and K. Williams, “A feature learning and object recognition framework for underwater fish images,” IEEE Transactions on Image Processing, vol. 25, no. 4, pp. 1862–1872, 2016.

[75] X. Li, M. Shang, H. Qin, and L. Chen, “Fast accurate fish detection and recognition of underwater images with fast r-cnn,” in OCEANS 2015-MTS/IEEE Washington. IEEE, 2015, pp. 1–5.

[76] S. Vicente, V. Kolmogorov, and C. Rother, “Graph cut based image segmentation with connectivity priors,” 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008.

[77] J. Carreira, and C. Sminchisescu, “Constrained parametric min-cuts for automatic object segmentation,” 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 2010.

[78] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[79] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, “Conditional random fields as recurrent neural networks,” In Proceedings of the IEEE International Conference on Computer Vision, pp. 1529-1537. 2015.

[80] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” Proceedings of the IEEE international conference on computer vision, 2015.

[81] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” In European conference on computer vision, pp. 740-755. Springer, Cham, 2014.

[82] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” In Proc. CVPR. 2017.

[83] C. Alippi, G. Boracchi, R. Camplani, and M. Roveri, “Detecting External Disturbances on Camera Lens in Wireless Multimedia Sensor Networks,” IEEE Transactions on Instrumentation and Measurement, vol. 59, pp. 2982-2990, 2010.

[84] V. Kanchev, K. Tonchev, and O. Boumbarov, “Blurred Image Regions Detection using Wavelet-based Histograms and SVM,” IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Prague, Czech Republic, pp. 457-461, 2011.

[85] T.-W. Huang, J.-N. Hwang, and C. S. Rose, “Chute based automated fish length measurement and water drop detection,” In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pp. 1906-1910. IEEE, 2016.

[86] R. Liu, Z. Li, and J. Jia, “Image partial blur detection and classification,” In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1-8. IEEE, 2008.

[87] R. Chan, H. Yang, and T. Zeng, “A two-stage image segmentation method for blurry images with Poisson or multiplicative gamma noise,” SIAM Journal on Imaging Sciences 7, no. 1 pp. 98-127, 2014.

[88] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3d traffic scene understanding from movable platforms,” IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 5, pp. 1012–1025, 2014.

[89] H. Zhang, A. Geiger, and R. Urtasun, “Understanding high-level semantics by modeling traffic patterns,” in Proceedings of the IEEE international conference on computer vision, 2013, pp. 3056–3063.

[90] S. Tang, M. Andriluka, B. Andres, and B. Schiele, “Multiple people tracking by lifted multicut and person reidentification,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3539–3548.

[91] A. Milan, K. Schindler, and S. Roth, “Multi-target tracking by discrete continuous energy minimization,” IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 10, pp. 2054–2068, 2016.

[92] S. Tang, B. Andres, M. Andriluka, and B. Schiele, “Multi-person tracking by multicut and deep matching,” in European Conference on Computer Vision. Springer, 2016, pp. 100–111.

[93] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele, “A multi-cut formulation for joint segmentation and tracking of multiple objects,” arXiv preprint arXiv:1607.06317, 2016.

[94] R. Kumar, G. Charpiat, and M. Thonnat, “Multiple object tracking by efficient graph partitioning,” in Asian Conference on Computer Vision. Springer, 2014, pp. 445–460.

[95] W. Choi, “Near-online multi-target tracking with aggregated local flow descriptor,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 3029–3037.

[96] S. Tang, B. Andres, M. Andriluka, and B. Schiele, “Subgraph decomposition for multi-target tracking,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5033–5041.

[97] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, “Multiple target tracking based on undirected hierarchical relation hypergraph,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1282–1289.

[98] A. Sadeghian, A. Alahi, and S. Savarese, “Tracking the untrackable: Learning to track multiple cues with long-term dependencies,” arXiv preprint arXiv:1701.01909, vol. 4, no. 5, p. 6, 2017.

[99] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie, “Trajectory factory: Tracklet cleaving and re-connection by deep Siamese bi-gru for multiple object tracking,” arXiv preprint arXiv:1804.04555, 2018.

[100] C. Kim, F. Li, and J. M. Rehg, “Multi-object tracking with neural gating using bilinear lstm,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 200–215.

[101] Y. Lu, C. Lu, and C.-K. Tang, “Online video object detection using association lstm,” in Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 2017, pp. 22–29.

[102] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, “Online multi-target tracking using recurrent neural networks.” in AAAI, vol. 2, 2017, p. 4.

[103] E. Ristani and C. Tomasi, “Features for multi-target multi-camera tracking and re-identification,” arXiv preprint arXiv:1803.10859, 2018.

[104] Z. Zhong, L. Zheng, D. Cao, and S. Li, “Re-ranking person reidentification with k-reciprocal encoding,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1318–1327.

[105] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Detect to track and track to detect,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3038–3046.

[106] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang et al., "T-cnn: Tubelets with convolutional neural networks for object detection from videos," IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[107] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 817–825.

[108] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," In 2010 IEEE computer society conference on computer vision and pattern recognition (pp. 3360-3367), IEEE, 2010.

[109] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823), 2015.

[110] J. Redmon, and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[111] J. Yang, K. Yu, Y. Gong, and T. S. Huang, "Linear spatial pyramid matching using sparse coding for image classification," In CVPR (Vol. 1, No. 2, p. 6), 2009.

[112] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," Journal of computer and system sciences, vol. 43, no. 3, pp. 425–440, 1991.

[113] M. Keuper, E. Levinkov, N. Bonneel, G. Lavou´e, T. Brox, and B. Andres, "Efficient decomposition of image and mesh graphs by lifted multicuts," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1751–1759.

[114] E. Levinkov, J. Uhrig, S. Tang, M. Omran, E. Insafutdinov, A. Kirillov, C. Rother, T. Brox, B. Schiele, and B. Andres, "Joint graph decomposition & node labeling: Problem, algorithms, applications," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6012–6020.

[115] S. A. A. Shah, U. Nadeem, M. Bennamoun, F. A. Sohel, and R. Togneri, "Efficient Image Set Classification using Linear Regression based Image Reconstruction," In CVPR Workshops (pp. 601-610), 2017.

[116] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects", In Computer Vision and Pattern Recognition (CVPR), IEEE Conference on (pp. 1201-1208), IEEE, 2011.

[117] C. Dicle, O. I. Camps, and M. Sznaier, "The way they move: Tracking multiple targets with similar appearance", In Proceedings of the IEEE international conference on computer vision (pp. 2304-2311), 2013.

[118] J. H. Yoon, M. H. Yang, J. Lim, and K. J. Yoon, “Bayesian multi-object tracking using motion context from multiple objects”, In Applications of Computer Vision (WACV), IEEE Winter Conference on (pp. 33-40), IEEE, 2015.

[119] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 3645-3649). IEEE.

[120] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking”, In European Conference on Computer Vision (pp. 17-35). Springer, Cham, 2016.

[121] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking”, arXiv preprint arXiv:1603.00831, 2016.

[122] G. Wang, J. N. Hwang, F. Wallace, C. Rose, “Multi-Scale Fish Segmentation Refinement and Missing Shape Recovery,” IEEE Access, 7, 52836-52845, 2019.

[123] G. Wang, J. N. Hwang, Y. Xu, F. Wallace, and C. S. Rose, “Coarse-To-Fine Segmentation Refinement and Missing Shape Recovery for Halibut Fish,” In 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 370-374). IEEE.

[124] G. Wang, J. N. Hwang, C. Rose, and F. Wallace, “Uncertainty-Based Active Learning via Sparse Modeling for Image Classification,” IEEE Transactions on Image Processing, 28(1), 316-329, 2019.

[125] G. Wang, J. N. Hwang, K. Williams, and G. Cutter, “Closed-Loop Tracking-by-Detection for ROV-Based Multiple Fish Tracking,” In 2016 ICPR 2nd Workshop on Computer Vision for Analysis of Underwater Imagery (CVAUI) (pp. 7-12). IEEE, 2016.