Counting Fish In Neptune Canada Video Feeds

David Rusk, Fred Song

I. Introduction

NEPTUNE Canada is an underwater ocean observatory located off the coast of Vancouver Island, British Columbia. A wide variety of data are being recorded, including video data which is made available to researchers over the internet through a web-based system. The large volume of data recorded precludes manual analysis. Automatic methods are desired to extract information which can then be used in higher level analysis.

This project focuses on videos from Barkley Canyon. Its goal is to count the number of Sablefish that appear in any given video segment. This goal requires computer vision algorithms that can detect and track fish in an uncontrolled, dynamic underwater environment.

II. Related work

Several publications exist which propose automated systems for counting fish. The systems described by Spampinato et al. (2008) [1] and followed up on by Nadarajan et al. (2011) [2] will be examined in more detail later in this report as they form the basis of the proposed approach. Nadarajan et al. (2009) [3] takes a different approach by incorporating "user-provided domain knowledge and heuristics used by image processing experts" to develop an automated system capable of tasks such as fish counting. Boom et al. (2012) [4] describes a system that takes a more machine learning oriented approach because it additionally tries to classify species of fish. It is a more computationally involved system which runs on supercomputers.

An endeavour closely related to Neptune is the Fish4Knowledge project [5] which is being developed in the European Union in order to study marine ecosystems and the effects of environmental changes. That project's main goal is to make the enormous amount of data being recorded available, and useful, to researchers. Specifically, automatic data extraction techniques are being developed which process the data into a database with an interface which researchers can use to "formulate and answer higher level questions".

III. Proposed Approach

A. Development Tools

OpenCV [6] is a popular open source computer vision library. We chose to use it for this project because it provides implementations of many fundamental algorithms, allowing development effort to concentrate on ways to put them together into an effective computer vision system. OpenCV currently offers C++, C, Python and Java interfaces. The Python bindings were chosen for this project because Python is dynamically typed and interpreted facilitating a faster development cycle. OpenCV and Python are free software and as such the resulting system is free for use by anyone.

B. Early Work

Segmentation of fish from the background proved to be a difficult problem; we experimented with many approaches that were unsuccessful. We first tried to use the Canny edge detector to find the outer contours of fish. However, we discovered this approach is too sensitive to movements in the background caused by dust and debris. Even small movements introduced significant noise in the resulting edges. Another approach attempted was averaging the first 50 frames without fish and using that as the background. The segmentation was then produced by subtracting this background from the current frame. This did not work very well either due to small variations in the background over time causing a lot of noise.

We also tried using optical flow as a means of segmenting the fish. We applied the dense optical flow algorithm of Farneback (2003) [7] to find the velocity of each pixel in the current frame. The frame is then segmented into a binary image with each pixel being moving or nonmoving based on whether its velocity is above a threshold value. Regions with large enough area were considered to be fish. Optical flow segmentation worked well for moving fish. Unfortunately, it proved to be computationally expensive and it failed when the fish was stationary.

Finally, we experimented with template matching using both the fish's eye and head. This did not work very well both because the fish could have any orientation and their eyes and head could be occluded or outside the frame.

C. Final System Overview

Eventually we decided that the approach suggested in [1] looked promising, so it would be attempted and then expanded on as needed. This approach involved two main steps: fish detection and fish tracking. The block diagram

in Fig. 1 gives an overview of the components of the final system, including both those presented in [1] as well as supplementary ones added. Each component shown will be discussed in the following sections.

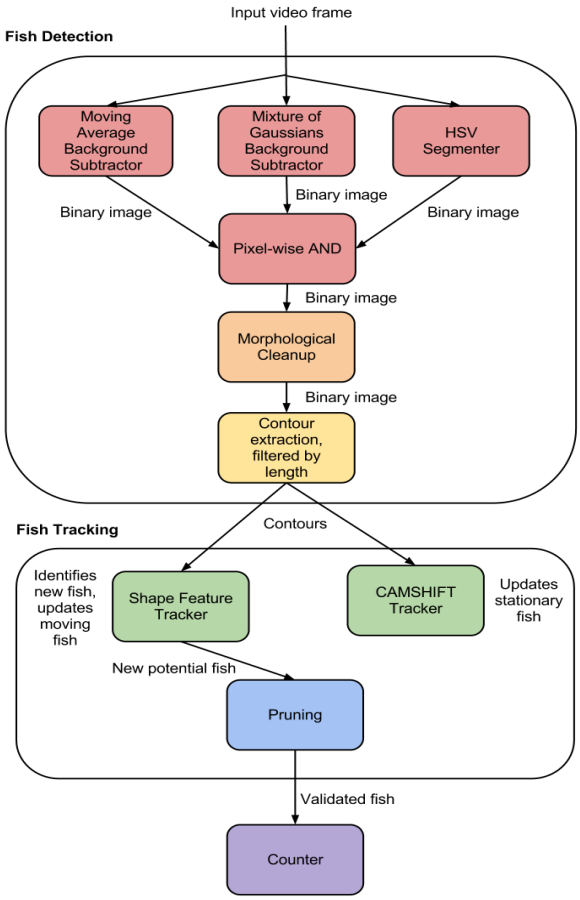

Fig. 1. Overall system architecture.

D. *Fish Detection*

Spampinato et al. [1] reported using a combination of two background subtraction methods in order to obtain binary segmentations of the fish. These methods were a moving average algorithm and an adaptive Gaussian mixture modelling algorithm (as described in [8]). Both provide a continually adapting model of the background which is appropriate for dynamic scenes where the background can be continuously changing.

1) *Moving Average Background Subtraction*

The moving average algorithm creates a model of the background (BG) in the current frame (CF) by taking the weighted sum of the background in the previous frame plus the current frame:

$$BG(i+1) = (1-\alpha)BG(i) + \alpha CF$$

The rate at which the background model adapts can be varied by using different values of alpha. The larger the value of alpha, the less weight is given to the history of the background, so it will update more quickly. The foreground can be determined by subtracting this background model from the current frame and selecting the pixels where the difference is above some threshold value.

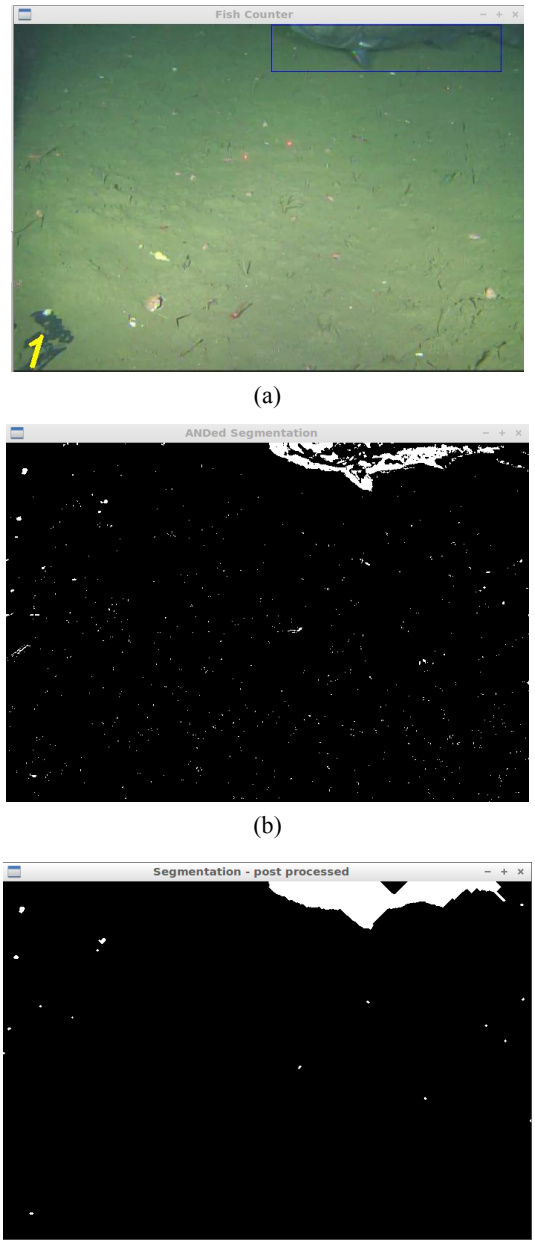

Fig. 2. a) Image frame, b) Pixel-wise AND of moving average and mixture of Gaussians background subtractors, c) Post-processed segmentation.

2) *Mixture of Gaussians Background Subtraction*

OpenCV has an implementation of the adaptive Gaussian mixture modelling algorithm of [8] (called BackgroundSubtractorMOG2 [9]), but as of this writing there is no Python binding. Therefore we selected a simpler version of the algorithm which did have a binding (BackgroundSubtractorMOG [10]). Essentially, this algorithm models pixels by a mixture of Gaussian distributions, with one for each colour. The algorithm then

determines the most probable colours. Pixels are classified as foreground if they are more than 2.5 standard deviations from any of the selected colour distributions.

3) Combining the Background Subtractors

These two background subtraction methods produced binary images which were combined through a pixel-wise AND operation. (A third segmentation method using the HSV colour space was added to remove shadows. It is discussed in section 5). The reason for using two algorithms is that the moving average algorithm has too many false positives due to small variations of dust particles and algae, which the mixture of Gaussians algorithm deals with better. The mixture of Gaussians algorithm is not used on its own because it has a tendency to fail on slowly moving objects, which the moving average algorithm helps correct.

4) Post-Processing

The segmentation produced by the background subtractors still has a fair bit of noise, and the fish have gaps and holes in them. These issues can be mitigated through a post-processing phase which uses morphological operators for clean up. An elliptical 3x3 structuring element is used for the processing. It was chosen to avoid the blocky appearance produced by square structuring elements. The processing consisted of 1 iteration of opening to remove many of the small objects, followed by 15 iterations of closing to fill in the fish. The results are shown in Fig. 2c.

5) Effect of Shadows

One problem encountered during testing of the detection system which was not addressed explicitly in [1] was dealing with shadows. Shadows have a large effect on the segmentation. As seen in Figure 3b the shadow is segmented as a large blob which is not easily distinguishable from a legitimate fish. The improved adaptable mixture of Gaussians algorithm [8] is supposed to perform some shadow removal. Perhaps this improved algorithm worked sufficiently for Spampinato et al. [1] Their example frames show the camera tilted much further upwards than in the Neptune data. This produces a greater depth of view, whereas the Neptune camera is pointing down at the ground and will therefore see more shadows in the same location that fish would be expected to occupy. This means that shadows may simply have not have been an issue for their target data set.

Using the simpler mixture of Gaussians algorithm, it is necessary to add an additional stage to handle shadow removal. The image is converted to the HSV colour space, and we experimentally found values representative of the fishes' texture. Pixels that have HSV values within the specified ranges are classified as fish. The resulting binary image is combined with the results of the background subtractors using the pixel-wise AND operation again. Fig. 3c shows the successful removal of the shadow from the final segmentation.

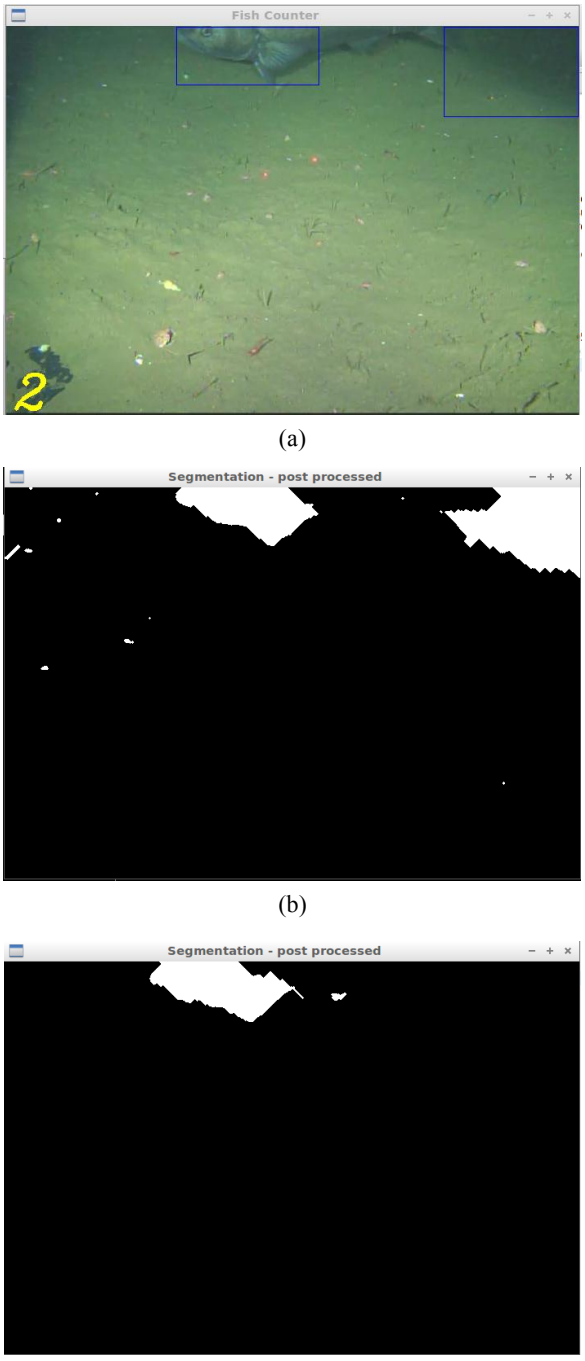

Fig. 3. a) Image frame, b) Shadows being segmented by background subtractors, c) Segmentation with shadow removed.

E. Fish Tracking

Fish tracking identifies the same fish across frames and picks up new fish that enter the scene. The previous stage of the system produced a binary segmentation of the fish. We used the contours of the regions in this binary image as the input to the tracking system. Note that contours with length under a certain threshold were filtered out because even after post-processing the segmentations sometimes had small noise objects.

Spampinato et al. [1] proposed using two tracking algorithms called a shape feature tracker and a CAMSHIFT tracker. Their paper is quite vague on how the trackers were combined, so a new method which plays to each algorithm's strength was developed for this project.

1) Shape Feature Tracker

The shape feature tracker uses three features to try and match fish between frames:

1. Centroid: the x and y coordinates of the centre of the fish's bounding box.
2. Area: the area within the contour of the fish.
3. Orientation: the angle of the minimum bounding box around the fish, with respect to the horizontal.

A fish in one frame would be matched to a fish in the next frame if the difference in these feature values is less than some threshold. The currently used thresholds are as follows:

1. If the centroid moves less than a Euclidean distance of 50 on the pixel grid, it matches.
2. If the area changes less than 40%, it matches.
3. If the orientation changes by less than 45 degrees, it matches.

These thresholds might seem quite high, but due to the variability in segmentation qualities it is necessary. For example, sometimes a fish's tail might be improperly segmented as a second object, greatly reducing its area. Even without poor segmentations, a fish with its broad side facing the camera could turn very sharply, causing a large change in area visible to the camera, and a large change in orientation between frames. Therefore the thresholds must be lenient, but this is acceptable since all three must be met reducing the chance of fish being incorrectly matched.

Notice that using a relative threshold for the area makes the algorithm more robust for working with varying fish sizes. If a specific threshold value is used, it would have to be tuned to a certain size of fish. A similar technique should be investigated for the centroid threshold taking into account fish sizes and perhaps the size of the field of view.

2) CAMSHIFT Tracker

CAMSHIFT stands for continuous adaptive mean shift. It performs tracking by leveraging colour information. First a region which is representative of the object to be tracked must be provided. A histogram of the hue channel is calculated for this region. Next a back projection of the target image is created by looking up the corresponding bin in the histogram for each pixel and using its value. This means that the back projection has high values for pixels whose colour is common in the object of interest. Another algorithm called mean shift then takes this back projection and the last known position of the object to find the most likely current position and size. The CAMSHIFT algorithm was originally proposed in [11] where more details of its operation can be found.

3) Combining the Trackers

The shape feature tracker worked well when fish were moving. However, it encountered trouble once fish stopped moving because they would gradually blend into the model of the background, causing them to be removed after background subtraction. On the other hand, the CAMSHIFT tracker worked quite well for stationary fish because it leveraged colour information that is available without movement. However, the CAMSHIFT algorithm had trouble tracking fish moving near each other. It would tend to expand the bounding box to incorporate multiple nearby fish, essentially merging them.

Therefore, to combine these tracking algorithms, we devised a new method not described in [1] to leverage the strengths of each algorithm. The system determines whether a fish is moving or not by recording at each frame the change in x and y coordinates of its bounding box centroid. If the difference between frames is near zero, the fish has stopped moving. Therefore it is possible to run a different tracking algorithm on a fish based on whether or not it is moving. The shape feature tracker is used for moving fish, while the CAMSHIFT tracker is used for stationary ones. The algorithm tracking a fish is free to change from frame to frame depending on the fish's movements; there is a continuous handoff.

F. Fish Counting and Pruning

Once a fish has been tracked, it is straightforward to increment a counter. However, we discovered in early results that there was a high degree of over counting, largely due to poor segmentations. For example, if a fish's tail becomes segmented independently from the rest of its body for a couple frames the fish could be counted twice. Several techniques were developed to mitigate this. They rely on the notion of a "potential object", which is an object on probation, meaning it has not been tracked very long and is subject to pruning. Pruning is used as a broad term encompassing the discarding of a potential object for any reason before it is counted as a verified fish.

The first method of pruning used is to simply discard objects which are tracked for less than a specified number of consecutive frames. This by itself eliminates many of the transient objects that result from poor segmentations. 15 frames is the current threshold. It could not be made too long or fish could enter and leave before being counted, and it could not be too short or the transient objects from poor segmentation might still exist.

A second form of pruning used is eliminating potential objects when they are found to have a high degree of overlap with a known fish. This is calculated using the bounding boxes, and is currently set so that if the overlap is more than 50% of the potential objects' bounding box area, it gets pruned.

Additionally, we assume that fish come into the field of view from the edge of the frame. This is reasonable since the camera points towards the sea bed. This allows objects that first appear somewhere in the interior of the frame to be pruned. Currently the interior of the frame is considered to be anything more than 20% of the way into the frame in any direction. This provides a bit of a buffer so that a fish

will not be prematurely pruned if there is a poor segmentation briefly when the fish enters the scene.

Once a potential object makes it through its probationary number of frames without being pruned away by one of these algorithms, it is considered a known fish and the counter is incremented.

G. System Output

In order to display the results of the system, the input video is annotated with bounding boxes around the currently tracked fish as well as a count of the total fish found so far. An example frame is shown in Figure 4.

Fig. 4. System output with counter and currently tracked fish.

In this example one fish has a red bounding box and the other has a blue one. They are colour coded based on the state of the fish. Red bounding boxes are for stationary fish being tracked by the CAMSHIFT tracker. Blue bounding boxes are for moving fish being tracked by the shape feature tracker. Yellow bounding boxes (not shown) are for potential fish which have yet to be counted.

IV. Experimental Results

A. Evaluation Criteria

The performance of the developed algorithm was tested by running it on videos acquired from the Neptune Canada online database [12]. A human observer watched each of these videos and then the developed algorithm was run on them. For each video, the following measurements were recorded:

1. The true fish count as determined by a human. This is considered to be the ground truth. Note that a fish is counted if over 30% of its body is in the scene.
2. The count determined by the algorithm.
3. The number of over-counted fish by the algorithm. This could be due to fish being counted more than once or non-fish objects being counted.
4. The number of fish missed by the algorithm.

These criteria give a more comprehensive evaluation of the performance beyond simply comparing the final count of the algorithm to the ground truth. This is because over-counts and undetected fish could end up cancelling each other out, hiding the poor performance.

B. Results

For the evaluation, 30 videos from the Barkley Canyon BC Mid-West POD4 Video Camera were used. The source videos were originally 5 minutes long, but included a camera pan in the middle. Therefore the original videos were divided into two videos each approximately 2 minutes long to avoid the pan. The following metrics can be derived from the evaluation criteria recorded:

correctly detected = algorithm count - algorithm over count
recall = correctly detected / ground truth count
precision = correctly detected / algorithm count

Table 1 contains the summary of the total results across the 30 evaluation videos.

Total Fish in database	153
Total Correctly Detected	74
Total False Detections	48
Total Missed	79
Total Recall	48.3%
Total Precision	60.1%

Table 1. Summary of evaluation results.

It should also be noted that 5 of the videos in the test set had no fish in them. The system worked correctly in each of these cases by determining a fish count of 0.

C. Discussion

Overall, our algorithm missed many fish leading to a recall of 48%, and also over counted many fish leading to a precision of 60%. The algorithm had high performance for simple scenes with one or two fish swimming normally across the frame. However, it is possible for the scene to become very complex with multiple fish overlapping each other, lots of dust, and partial occlusion of the camera by fins (Fig. 5). In these cases the algorithm failed badly.

The aggressive segmentation and pruning resulted in many fish being undetected. Due to the HSV segmenter, the segmented images of many fish were too small to pass the contour length threshold needed for them to be tracked. Occasionally the region would grow big enough, but the frame count and edge pruning caused the tracker to ignore it since it is not big enough for a long enough duration or it appeared in the middle of the scene. Reducing these threshold values to improve accuracy is tricky since that also increases the number of objects incorrectly determined to be fish, which lowers precision.

Another issue is that after a scene with many fish, the background subtraction (specifically the Mixture of Gaussians algorithm) will have trouble detecting fish for a period afterwards (Fig. 6).

Over counting of fish was fairly common. Many times, especially when the fish moved very slowly or stopped, the segmented fish would be split into more than 1 region causing the tracker to detect it as two or more different

fishes (Figure 7). This was also caused by the background subtraction as parts of the fish blend into the background.

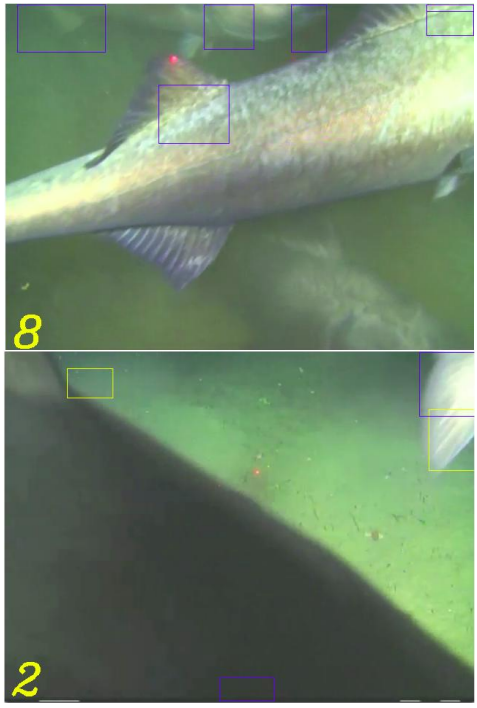

Fig 5. Complex scenes: dust and overlapping fish (top), blocking of camera (bottom)

An area where the system did perform quite well was in avoiding false detections due to non-fish objects such as shadows, eels and jellyfish. This was largely due to the HSV segmentation which leveraged colour information.

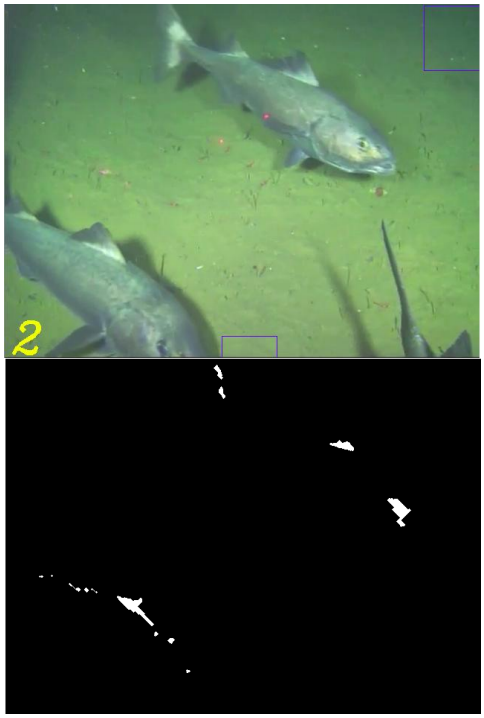

Fig. 6. Insensitive background subtraction after scene with many fish.

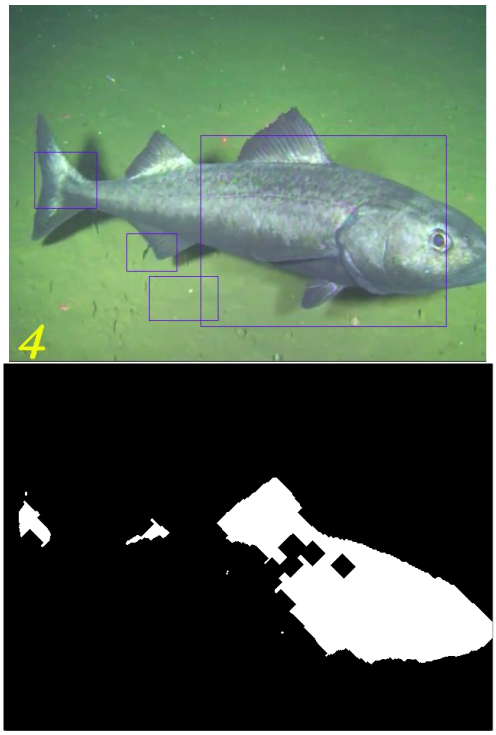

Fig. 7. Incorrect segmentation of fish leading to over counting

V. Conclusions

The system developed performs well in fairly simple scenarios. A simple scenario can be considered one where there are between one and three fish on the screen at a given time and the fish do not overlap with each other. More work is required to achieve good results in more complex scenarios where there is occlusion and the field of view is mostly filled with fish.

Currently, the segmentation and pruning is quite aggressive resulting in many fish not being detected and fishes being over-detected. The performance of the segmentation algorithms as well as the shape feature tracker could be improved by tuning their parameters. Even better would be to develop methodologies for calculating good parameters automatically, which would allow the algorithms to automatically adapt to changes in illumination or resolution between videos.

Visually the textures of the fish are highly distinctive from that of the background. Therefore, future work might also investigate the use of texture analysis coupled with region growing to aid in the segmentation stage.

Acknowledgements

Investigation of edge-based algorithms: David Rusk and Fred Song
Investigation of optical flow: David Rusk and Fred Song
Investigation of template matching: Fred Song
Implementation of final system segmentation algorithms: David Rusk
Implementation of final system tracking and pruning algorithms: David Rusk
Evaluation of final system: Fred Song

REFERENCES

[1] C. Spampinato, Y.H. Chen-Burger, G. Nadarajan, R. Fisher, "Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos," *Proc. 3rd Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, vol. 2, pp. 514-519, 2008.

[2] G. Nadarajan, Y.H. Chen-Burger, R.B. Fisher, C. Spampinato, "A flexible system for automated composition of intelligent video analysis," *Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium on* , vol., no., pp.259,264, 4-6 Sept. 2011.

[3] G. Nadarajan, Y.H. Chen-Burger, R. Fisher, "A Knowledge-Based Planner for Processing Unconstrained Underwater Videos", *IJCAI workshop on Learning Structural Knowledge from Observations*, 2009.

[4] B. Boom et al., "Long-term underwater camera surveillance for monitoring and analysis of fish populations" *Proc. of Int. Workshop on Visual observation and Analysis of Animal and Insect Behavior (VAIB), in conjunction with ICPR 2012, Tsukuba, Japan*, 2012.

[5] Fish4Knowledge Project Homepage [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/Fish4Knowledge/

[6] OpenCV Homepage [Online]. Available: http://opencv.org/

[7] Gunnar Farneback, "Two-frame motion estimation based on polynomial expansion", Lecture Notes in Computer Science, 2003, (2749), 363-370.

[8] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction", *International Conference Pattern Recognition*, UK, August, 2004.

[9] OpenCV, BackgroundSubtractorMOG2. Available: http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html?highlight=backgroundsubtractormog#backgroundsubtractormog2

[10] OpenCV, BackgroundSubtractorMOG. Available: http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html?highlight=backgroundsubtractormog#backgroundsubtractormog

[11] Bradski, G.R. "Computer Vision Face Tracking for Use in a Perceptual User Interface", Intel, 1998.

[12] Neptune Canada Data Search [Online]. Available: http://dmas.uvic.ca/home