2nd Workshop on Computer Vision for Analysis of Underwater Imagery

Adaptive Foreground Extraction for Deep Fish Classification

Nicole Seese
Computer Science
Millersville University of Pennsylvania
Millersville, Pennsylvania

Andrew Myers, Kaleb Smith
and Anthony O. Smith
Electrical and Computer Engineering
Florida Institute of Technology
Melbourne, Florida

***Abstract*—Despite the recent advances in computer vision and the proliferation of applications for tracking, image classification, and video analysis; very little applied work has been done to improve techniques for underwater video. Object detection and classification for underwater environments is critical in domains like marine biology, where scientist study populations of underwater species. Most applications assume either a static background, or movement that can be accounted for by some constant offset. Existing state-of-the-art algorithms perform well under controlled conditions, but when applied to underwater video of an unconstrained real world environment, they suffer a substantial performance degradation. In this work, we implement a system that performs foreground extraction on streaming underwater video for fish classification using a convolutional neural network. Our goal is to accurately detect and classify objects in real-time utilizing graphics processing unit (GPU) parallel computing capability. GPU accelerated computing is the ideal hardware technology for video analysis that provides a platform for real-time processing. We evaluate our performance on standard benchmark video datasets, specifically for scene complexity, and for detection and classification accuracy.**

***Index Terms*—background estimation, object detection, deep neural networks**

I. Introduction

The computer vision community has made remarkable advances in video processing research. In fact, computer vision applications have garnished enough momentum to play a fundamental role in every-day activities. For underwater processing, particular attention is given to object detection which largely consist of isolating moving objects in a video sequence. There is a lot of hesitation to explore research in this domain, due to the fact that underwater video introduces challenges that are not conventional in traditional video analysis.

Underwater videos contain dynamic scenes with one or more of the following properties that impact quality and pose major concerns for algorithms. For example, variable illumination from sunlight is refracted differently and has the potential to introduce glare to the capture device. Motion caused by waves or current can impose gyration of the platform. Also, natural obstructions such as silt, sediment, and bubbles can be problematic. Light variation from the natural sunlight during certain times, affects the refraction of the light in water. This work proposes a system with 3 main considerations: 1) robust object detection, 2) accurate classification, and 3) real-time processing capability.

Our framework begins by generating independent background models that are amalgamated to compose a single robust model. For each frame n_i , where $i = \{1, 2, \dots, N\}$, a shape contour is obtained for all identified objects, see fig (1) for an illustration of our overall process. Once the object is detected a convolutional neural network (CNN) performs automated feature extraction and classification.

The remainder of the paper is organized as follows. We will begin with a brief discussion of related work in underwater object detection, Section II. This is followed by Section III where we provide a detailed description of generating a robust background model. Next, an overview of TensorFlow is discussed in Section IV. Experimental results in Section V showcase our performance on a number of standard benchmark datasets. Section VI concludes with a summary discussion for future research.

II. Related Work

In a survey by Kavasidis and Palazzo [6], they evaluated the performance of several state-of-the-art algorithms on unconstrained underwater videos. The algorithms Video Background Extraction [2], Adaptive Poisson Mixture Model [3], CodeBook [8], Wave-back [14], Intrinsic Model [13], and Gaussian Mixture Model [17] were evaluated on a set of expert labeled object detection videos. They concluded that the algorithms performed well on videos that contained clear water scenes and uniform backgrounds. When phenomena were present, the performance of all the algorithms, in detecting objects, degraded to the point where they were declared unusable.

For nearly two decades Kalman filtering has been used as a means of adaptive foreground extraction. The original work by Ridder et al. [16], considered varying illumination, but assumed a static closed captioned digital camera which is typically used for surveillance. More recently, the work by Lei and Zhao [10], extended this technique to underwater video and applied it to a solution for monitoring the bottom of a swimming pool. They also assumed a stationary monitoring camera, and in order to establish a mathematical model, the initial background had to be captured when the pool is completely unoccupied. This is not feasible in an unconstrained environment such as a lake, river, or open ocean.

Some promising research by [15], proposed sparse and low-rank matrix decomposition as a new method for foreground

978-1-5090-5870-9/16 $31.00 © 2016 IEEE
DOI 10.1109/CVAUI.2016.22

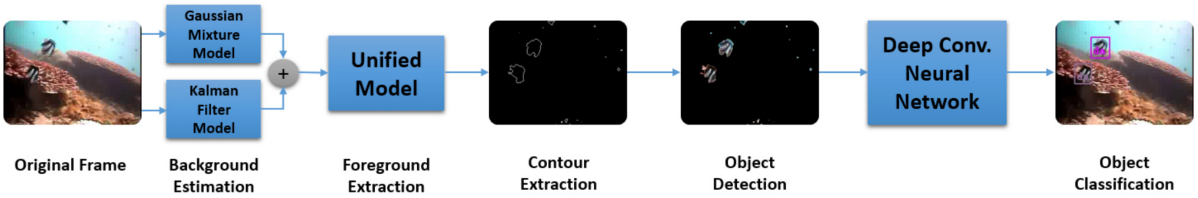

Fig. 1. Underwater video processing overview. Our approach processes each frame to generate adaptive independent (left) background models. The background model is continuously updated with the most current frame information; then shape contours and object image extraction occur (center). Finally (right), the extracted image is processed by the convolutional neural network (CNN), which automatically generates features for classification.

extraction. Given a N frame video sequence, where each frame is represented as a $m \times 1$ column vector, they construct a $M \in \mathbb{R}^{m\times n}$ matrix for the full video sequence. This approach has demonstrated good results, but relies on all N frames to be available for processing. Also, for a relatively large number of frames this would be computational and memory expensive, making this approach not practical for neither continuous video stream nor real-time processing.

Deep learning is a new area of machine learning research with the objective of moving the science closer to one of its original goals of artificial intelligence. Convolutional deep neural networks are used to learn multiple levels of representation and abstraction that help to make sense of data. The excitement about deep learning is largely in the application of unsupervised feature learning and the use of deep networks. A tremendous amount of work has been performed in this area, but our work will focus specifically on the use of TensorFlow [1] by Google Research. The TensorFlow project is a deep learning library for which the fundamental idea of the framework is to construct a data flow graph that defines computations. Nodes represent operations, and edges represent tensors, which are multidimensional arrays. The entire data flow graph is a complete description of computations, which occur with a TensorFlow session, and are executed on either the *CPU* or *GPU*.

III. Background Model Estimation

Background estimation is fundamental to detecting moving objects in video. Many detection algorithms use background estimation techniques to model the observed environment, but this becomes a challenge when attempting to model dynamic scenes. During surveillance of underwater environments, moving backgrounds (waving plants, clutter) and illumination changes (weather changes, reflections, etc.) are major obstacles for background model estimation. The development of a single algorithm that overcomes these challenges is usually not feasible. Several approaches have been proposed for generic model estimation and change detection, with some of the historical work being [9], [12]. In this work, we focus on underwater video with unknown illumination parameters, dynamic backgrounds, and a non-static imaging platform. For this reason, we elect two algorithms that enable a more robust system: Gaussian mixture model and Kalman filtering. Each algorithm computes an independent model estimation for both foreground and background, then the foregrounds are fused to complement the advantages and provide a robust object detection.to provide robust detections

We adopt a fusion approach that generalizes background estimation into two categories; long-term model and short-term model [11]. The long-term model approach performs background estimation over a longer period of time, therefore relying on more frames to adapt an estimation. The contrary short-term model approach adapts an estimation relatively fast with fewer frames.

A. Parametric Model Estimation

The Gaussian distribution was first used for background estimation by Wren et al. [18], where they modeled small movements using a single Gaussian for the entire frame. For diverse scenes this is not sufficient because within a given time period t, there are many intensity changes for each pixel. To be more robust, in a given frame each pixel can be modeled by a single Gaussian (where a pixel model is described by the mean intensity μ, and variance σ) over time. To model more dynamic backgrounds a Gaussian mixture model (GMM) was proposed in [4], and made more efficient in work by Stauffer et al. [17]. The approximation of complex distributions by the GMM algorithm with several Gaussians is computational intensive, but we overcome this hurdle by utilizing parallel computing hardware and software techniques. The GMM computes robust long-term models of the background capable of emphasizing situations with slow moving objects. The algorithm allows for multi-modal background estimations where the same pixel can have different background models.

The first step in the parametric model estimation is to extract the current frame from a video stream and convert the pixels to grayscale for simplicity. After converting the image into grayscale, the pixels are separated, and each GMM is initialized with K Gaussian distributions. When estimating simple backgrounds, a choice of K in a range $3 \leq K \leq 5$ is sufficient; however a higher K is desirable for video with complex backgrounds so that information can be retained and used to estimate background pixels in the subsequent frames. A default value of $K = 5$ was used in our experiments which produced favorable results in both simple and complex backgrounds. The current pixel value $X_t = f(x, y)$, is then compared to the current distribution k, given observations from time 1 through t. The μ and σ values for unmatched distributions remain the same, but if the value is within $\mu - 2.5\sigma \leq X_t \leq \mu + 2.5\sigma$ deviations of the Gaussian's mean; then we flag a match. When a match is identified, the μ and σ of the Gaussian are updated by the following eq (1) and eq (2) respectively.

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \tag{1}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{2}$$

where

$$\rho = \alpha\eta\left(X_t|\mu_k, \sigma_k\right). \tag{3}$$

Here α is a constant that determines the learning rate. If the matched distribution for the pixel is considered a background model, then the pixel is background and the same strategy to label a foreground pixel.

B. Predictive Model Estimation

The short-term algorithm we elected was Kalman filter estimation [10], due to its ability to adapt to dynamic backgrounds, as well as its overall simplicity of calculation. A separate Kalman filter is applied to each of the $r \times c$ pixels of each N frame of the video stream; with the state modeled as the background intensity for each pixel. In this algorithm the first frame is taken as the initial predicted background. Next, the pixel predicted intensity is compared with its actual measured intensity to determine the difference between the predicted Kalman filter intensity and the actual pixel value. If the intensity is greater than a pre-determined threshold T, the pixel is labeled foreground; else the pixel is labeled background. Our system used a threshold value of $T = 25$, a value representing approximately 10% of the available $[0 - 255]$ range of pixel intensities. This value was obtained as a result empirical experiments, and visual error inspections.

Our threshold allows for slight variations in the background, such as changes in illumination, to still be recognized by the system as background; however, if a great difference occurs, it is assumed that a foreground object has moved into the frame and is now covering that pixel. The model at each pixel based on Kalman filter is shown in

$$\begin{bmatrix} X_{t+1} \\ \Delta X_{t+1} \end{bmatrix} = A \begin{bmatrix} X_{t+1} \\ \Delta X_{t+1} \end{bmatrix} + K\left[I_t\left(x, y\right) - X_t\right]. \tag{4}$$

Here, A is the system state matrix indicating the background dynamics. X is the estimation value of a series of frames at pixel (x, y). ΔX is the change in the background for that point when a new frame proceeds. I is the input image and K is the Kalman gain. This process is repeated for each frame, therefore allowing the predicted background to evolve as the background changes in the underwater video.

C. Model Fusion

The algorithms mentioned in Section III-A and Section III-B can be fused to complement the advantages and provide a more robust model estimation. We employ a similar background fusion approach as described in [11]. They utilize a rule based method that defines specific actions depending on the pixel outcome of independent models; neither, one, or both algorithms label a pixel as foreground. In our investigation we found that a simple intersection of the models gave best results. For performance improvements we create a binary pixel mask from the labeled output of the parametric and predictive models. The pixel mask allows for element-wise multiplication with its respective video frame to extract relevant foreground information. Our fused binary scheme labels a pixel as foreground if both algorithms agree; else it is regarded as background.

Fusion aims to address the specific deficiencies associated with the individual algorithms. First, the Kalman filter only maintains a single background model, eventually it begins to incorporate objects that remain in a stationary position for an extended period of time. Also, if an object has been incorporated into the background, then begins to move, the background pixels at that particular location will falsely register as foreground. The GMM often produces an abundance of isolated pixel noise in its background estimation due to the number of samples needed to provide an accurate estimation of the distribution. Because the two algorithms exhibit significantly different issues, a pixel-wise intersection of their binary foreground masks can often create an ideal foreground extraction.

D. Real-Time Implementation

The running time of an algorithm increases with the input size, although this may vary for different inputs of the same size. A common implementation of a video processing algorithm is is done with nested loops where the outer loop over the number of frames N, the second level loop iterates over rows (r), and the inner loop iterates over columns (c). This implies that the standard video analysis implementation is characterized as running in $O\left(N \times r \times c\right)$ time complexity. This can be drastically reduced by taking advantage of GPU programming designed for vector processing applications. On these hardware devices each pixel can be processed independently, which is the ideal for data parallelism. We implemented critical components of our system in CUDA on a NVIDIA GPU. To process a frame we launch a CUDA kernel that generates a massive number of threads to exploit data parallelism. We developed parallel processing code for both the Kalman and GMM algorithms. This permits our system to perform in near real-time reducing our overall running time to approximately $O\left(N\right)$ complexity, only dependent on the total number of frames. Our experiments on HD video streams demonstrated an average processing rate of > 25 frames per second.

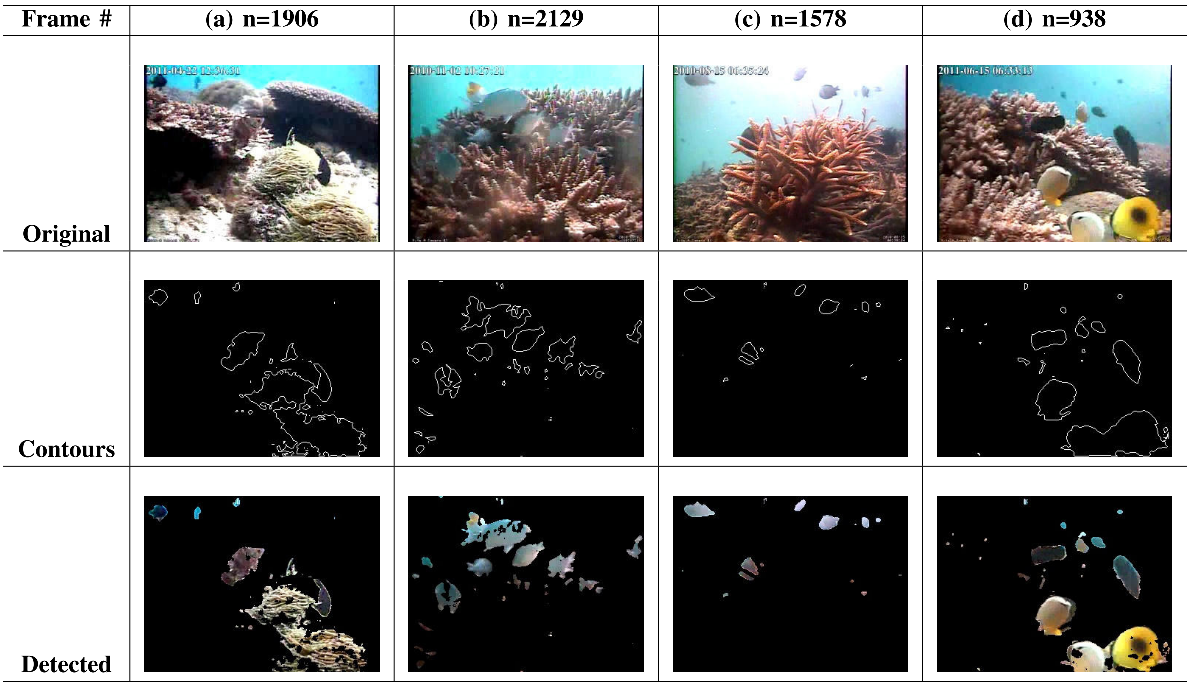

Fig. 2. Object detection results. **(a)** The *Dynamic Background,* features background object movement. **(b)** *Luminosity Changes* highlight abrupt lighting changes. **(c)** *Camouflage Foreground Objects* emphasize the effects of camouflage. **(d)** *Hybrid* shows a combination of the aforementioned conditions.

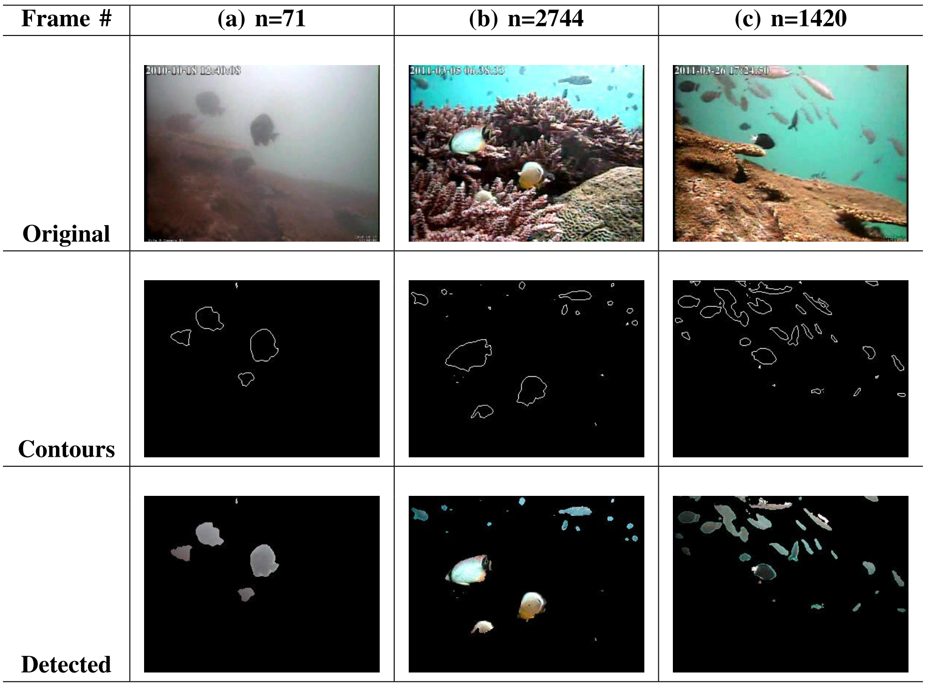

Fig. 3. Object detection results. **(a)** *Blurred* illustrates smoothed and low contrasted images. **(b)** *Complex Background* shows where background is featured by complex textures. **(c)** *Crowded* with many occluding objects.

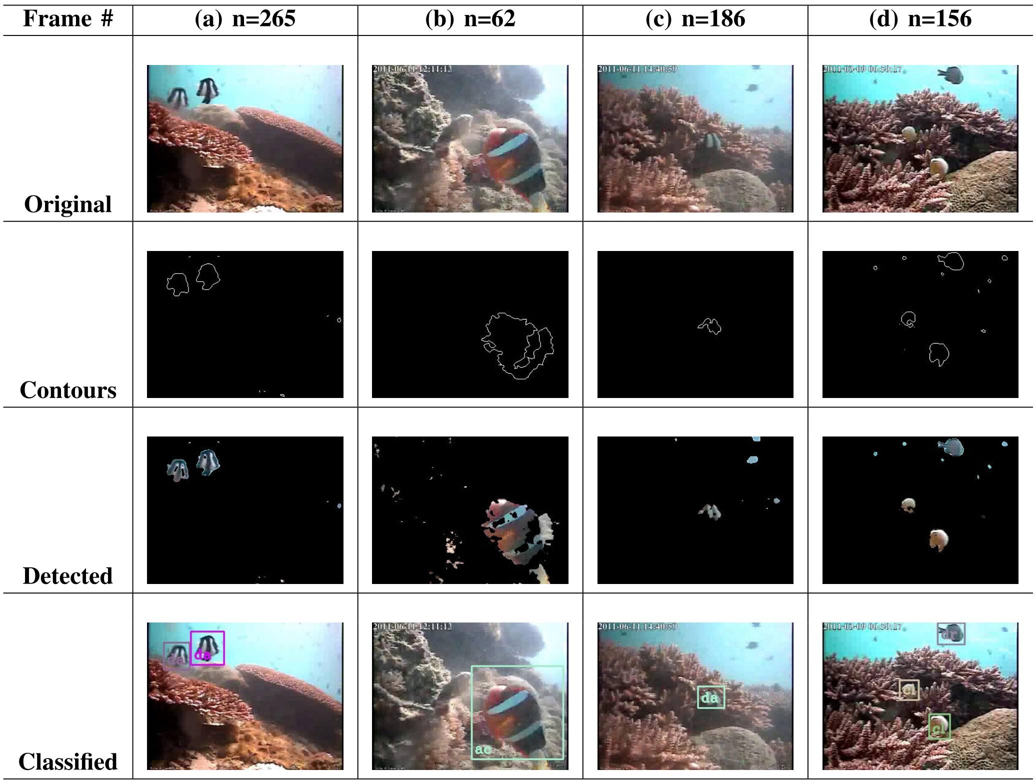

Fig. 4. Classification results. Unique video identification names as they relate to the SeaCLEF dataset. These names apply to the experiments in; **(a)** sub_0a3548f4df96ac98d7f226aa1125fd06#201103090650_0 **(b)** sub_7edc66df98960ea4057a349575729f86#201110081020_6 **(c)** sub_b4c5f7325863067e5179e21fee6484e3#201106111440_3 **(d)** sub_e830f103a30f6890a84bee30b31ae490#201106111210_1

IV. Feature Learning and Classification

This framework includes automated feature learning and classification with TensorFlow, Google's open-source numerical computation library [1]. TensorFlow is a CNN library that offers a C++ API which allows for easy integration, and support for both CPU and GPU execution. The TensorFlow library accomplishes tasks through the use of data flow graphs, on which nodes can represent both mathematical operations or I/O processes, and directed edges represent the flow of data stored in tensors or multi-dimensional arrays. Although, the framework can be used in a variety of applications, these data flow graphs that compose its core functionality allow for it to easily extend to building, training, and testing neural networks.

For our work we focused on fish classification, training the neural network with the appropriate classes. In order to classify individual fish, we segmented the foreground extracted objects. However, the objects of interest were not always the sole portions of the extracted foreground. Artifacts such as debris, bubbles, or camera glare may be present, and should be removed. Finally, the segmented portion of the frame is processed by the CNN for automated feature extraction and classification.

V. Experimental Results

Since our system includes both background estimation and classification we found it appropriate to evaluate the effectiveness of our proposed method on two benchmark datasets. Initially, for robust foreground estimation we tested using the Underwater Benchmark Dataset For Target Detection Against Complex Background[1] [7]. This benchmark dataset consists of 7 categories and represents complex challenges in underwater video background modeling. The seven categories are Blurred, Complex Background, Crowded, Dynamic Background, Hybrid, Camouflage Foreground Object, and Luminosity Variations, respectively.

For evaluation we performed experiments to address the more complex cases. Results in fig (3) and fig (2) provide a qualitative measure of how fusing the long-term and short-term background models allowed us to extract relevant foreground objects. We selected a subset of frames that best reflects the theme of the respective videos provided in the dataset. On the tested videos we observed promising results for foreground extraction. We were able to extract a substantial number of objects per frame in all the benchmark videos.

[1] All of the videos for this dataset are downloaded from http://f4k.dieei.unict.it/datasets/bkgmodeling/.

The second dataset was SeaCLEF 2016[2][5] and used to evaluate classification accuracy. SeaCLEF 2016 visual dataset contains both videos and images of marine organisms. The data is to be used to help identify underwater species from video. The training data consists of 20 videos, 15 fish species, a set of training images for each species. The fish are labeled in each video for ground truth, where the labels consists of bounding boxes (one for each instance of the given fish species) together with the fish species. The test data includes 73 underwater videos.

In fig (4) we show 4 samples that include a variety of obstacles which are reflective of the seven obstacle categories. These were chosen from the test data provided with SeaCLEF 2016 resources. In addition to the foreground detected objects, we include the TensorFlow classification in the lower row of fig (4). With high confidence, TensorFlow was able to accurately classify multiple fish within a frame, even when there was considerable background noise.

VI. Conclusion

Our overall objective was to produce a robust system with the capability to classify multiple objects in real-time from underwater video. We presented a background estimation strategy that fused a predictive Kalman filter, and a parametric Gaussian mixture model algorithm to combine the strengths of both algorithms. We implemented our strategy using heterogeneous parallel computing techniques that utilized massively parallel threads to achieve improved running times when compared to traditional serial implementations.

In this work we incorporated a deep convolutional neural network for automated feature extraction and classification to construct and end-to-end system. The system was evaluated on benchmark datasets where we provided qualitative results of the algorithms performance. We demonstrated the robustness of the methods by handling object detection and classification on complex video scenes and making no prior assumptions. In future work we plan to investigate various other features and continue to explore potential run time performance improvements.

Acknowledgment

The authors acknowledge partial support from NSF grants Nos. 1263011 and 1560345. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. ManÃ©, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems. Technical report, 2015. Software available from tensorflow.org.

[2] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, 2011.

[3] Alberto Faro, Daniela Giordano, and Concetto Spampinato. Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1398–1412, 2011.

[4] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 175–181. Morgan Kaufmann Publishers Inc., 1997.

[5] Alexis Joly, HervÂŽe Goeau, HervÂŽe Glotin, Concetto Spampinato, Pierre Bonnet, Willem-Pier Vellinga, Julien Champ, Robert PlanquÂŽe, Simone Palazzo, and Henning Muller. Lifeclef 2016: Multimedia life species identification challenges. Technical report, 2016.

[6] Isaak Kavasidis and Simone Palazzo. Quantitative performance analysis of object detection algorithms on underwater video footage. In *Proceedings of the 1st ACM international workshop on Multimedia analysis for ecological data*, pages 57–60. ACM, 2012.

[7] Isaak Kavasidis, Simone Palazzo, Roberto Di Salvo, Daniela Giordano, and Concetto Spampinato. An innovative web-based collaborative platform for video annotation. *Multimedia Tools and Applications*, 70(1):413–432, 2014.

[8] Kyungnam Kim, Thanarat H Chalidabhongse, David Harwood, and Larry Davis. Background modeling and subtraction by codebook construction. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3061–3064. IEEE, 2004.

[9] B Lee and M Hedley. Background estimation for video surveillance. In *Image and vision computing New Zealand*, pages 315–320, 2002.

[10] F. Lei and X. Zhao. Adaptive background estimation of underwater using kalman-filtering. In *Image and Signal Processing (CISP), 3rd International Congress on*, volume 1, pages 64–67, Oct 2010.

[11] E. Monari and C. Pasqual. Fusion of background estimation approaches for motion detection in non-static backgrounds. In *Advanced Video and Signal Based Surveillance, IEEE Conference on*, pages 347 – 352, Sept 2007.

[12] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[13] Fatih Porikli. Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 20–27. IEEE, 2005.

[14] Fatih Porikli and C Wren. Change detection by frequency decomposition: Wave-back. In *Proc. of Workshop on Image Analysis for Multimedia Interactive Services*, 2005.

[15] Hongwei Qin, Yigang Peng, and Xiu Li. Foreground extraction of underwater videos via sparse and low-rank matrix decomposition. In *Computer Vision for Analysis of Underwater Imagery (CVAUI), 2014 ICPR Workshop on*, pages 65–72. IEEE, 2014.

[16] C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Recent Advances in Mechatronics, Int. Conf. on*, pages 193–199, 1995.

[17] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 2, June 1999.

[18] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):780–785, 1997.

[2] The videos for the SeaCLEF dataset are downloaded from http://www.imageclef.org/lifeclef/2016/sea.