Using Computer Vision And Volunteer Computing To Analyze Avian Nesting Patterns And Reduce Scientist Workload

Kyle Andrew Goehner

USING COMPUTER VISION AND VOLUNTEER COMPUTING TO ANALYZE AVIAN NESTING PATTERNS AND REDUCE SCIENTIST WORKLOAD

by

Kyle Andrew Goehner
Bachelor of Science, University of North Dakota, 2013

A thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota
August
2015

This thesis, submitted by Kyle Andrew Goehner in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work as been done and is hereby approved.

[illegible]
Dr. Travis Desell

[illegible]
Dr. Ron Marsh

Wen-Chen Hu
Dr. Wen-Chen Hu

Susan N Ellis-Felege
Dr. Susan Ellis-Felege

This thesis is being submitted by the appointed advisory committee as having met all of the requirments of the School of Graduate Studies at the University of North Dakota and is hereby approved.

Wayne Swisher
Wayne Swisher
Dean of the School of Graduate Studies

May 21, 2015
Date

PERMISSION

Title	Using Computer Vision and Volunteer Computing to Analyze Avian Nesting Patterns and Reduce Scientist Workload
Department	Department of Computer Science
Degree	Master of Science

Kyle Andrew Goehner
August 2015

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost I would like to thank God for my health and making this work possible. I am also incredibly thankful for my partner, Hannah Muhs, and her continuous encouragement, mental and emotional support.

I want to thank my thesis committee for their time, patience, and contribution to my development as a scientist. I am particularly thankful for Dr. Susan Ellis-Felege and her team of biologists for providing video and classification data, without them this research would not have been possible.

Most of all, I wish to express my sincere appreciation to my thesis advisor, Dr. Travis Desell. The expertise and knowledge he has shown will prove invaluable in my future works. He has provided an excellent work environment and constantly encourages development and growth. His enthusiasm for research and computing is truly inspiring.

Finally, I want to thank the Wildlife@Home citizen scientists for their support and dedication. They have spent significant amounts of time watching videos and providing feedback for the project. Their participation is paramount to this research and the Wildlife@Home project as a whole.

This work has been partially supported by the National Science Foundation under Grant Number 1319700. 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 National Science Foundation. Funds to collect data in the field were provided by the U.S. Geological Survey, North Dakota Game and Fish, and North Dakota Faculty Seed Grant.

ABSTRACT

This paper examines the use of feature detection and background subtraction algorithms to classify and detect events of interest within uncontrolled outdoor avian nesting video from the Wildlife@Home project. We tested feature detection using Speeded Up Robust Features (SURF) and a Support Vector Machine (SVM) along with four background subtraction algorithms — Mixture of Guassians (MOG), Running Gaussian Average (AccAvg), ViBe, and Pixel-Based Adaptive Segmentation (PBAS) — as methods to automatically detect and classify events from surveillance cameras. AccAvg and modified PBAS are shown to provide robust results and compensate for issues caused by cryptic coloration of the monitored species. Both methods utilize the Berkeley Open Infrastructure for Network Computing (BOINC) in order to provide the resources to be able to analyze the 68,000+ hours of video in the Wildlife@Home project in a reasonable amount of time. The feature detection technique failed to handle the many challenges found in the low quality uncontrolled outdoor video. The background subtraction work with AccAvg and the modified version of PBAS is shown to provide more accurate detection of events.

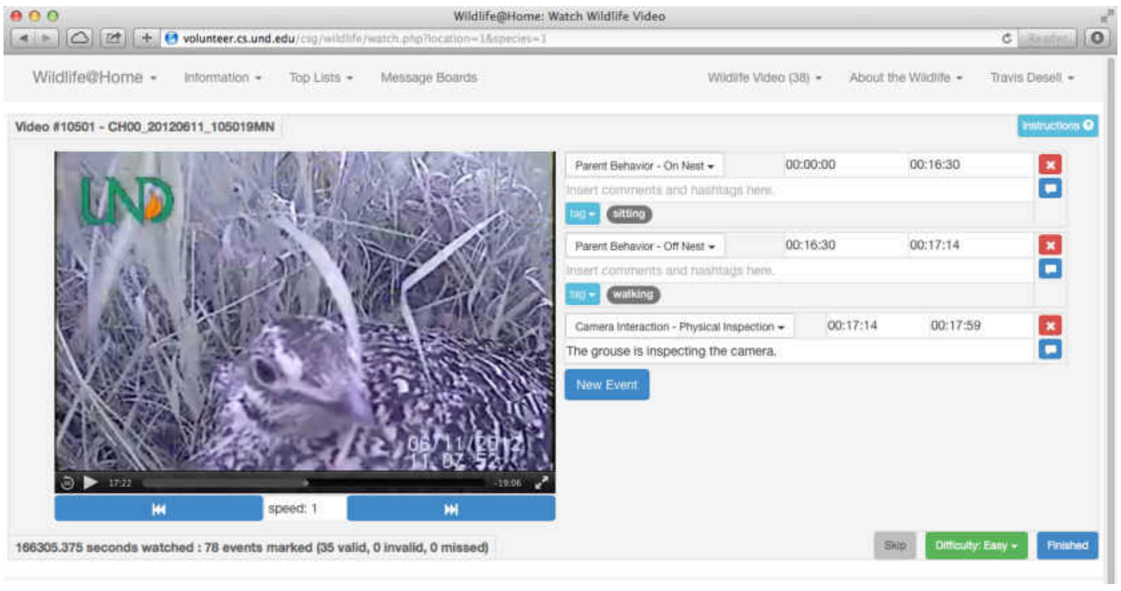
CHAPTER I
INTRODUCTION

Wildlife@Home[1] [1, 2] is a volunteer computing project in which citizen scientists and wildlife experts are presented videos at the nests of various species of birds. Currently, users have the option of viewing Sharp-Tailed Grouse (*Tympanuchus phasianellus*, an *indicator species* which can represent ecological health), Interior Least Tern (*Sternula antillarum*, a federally endangered species), or Piping Plover (*Charadrius melodus*, a federally threatened species). Each of these species have different nesting behaviors and users are tasked with classifying them. Examples of behaviors are *On Nest*, *Off Nest*, *Brooding*, *Flying*, *Foraging*, and *Feeding*. While users are observing the nests, they create a set of events for each video specifying when the events begin and end. Each event in has a type, start time, and end time (see Figure 1).

Such camera studies are popular in the field of avian ecology as they can reduce researcher impacts on animal behavior and also monitor animals in remote locations [3, 4]. Unfortunately, many of these studies are hampered by small sample sizes, where few have studied more than 100 nests [4], limiting the biological inferences that can be made. In order to overcome these challenges, Wildlife@Home has been developed to employ both volunteer computing and crowd sourcing to quickly analyze wildlife video, as well as to investigate automated video analysis strategies using computer vision techniques.

The Wildlife@Home project has accumulated over 85,000 hours of 24/7 uncontrolled outdoor surveillance video. This amount of data becomes problematic for humans to classify, even with software tools to help create and store event data. A lot of time is spent viewing regions of the video where the birds are not present at all or where a bird is present but highly inactive for long periods of time. Users watching video can use the scrub bar to move more

[1]http://volunteer.cs.und.edu/csg/wildlife/



Video #10501 - CH00_20120611_105019MN

Parent Behavior - On Nest	00:00:00	00:16:30
Parent Behavior - Off Nest	00:16:30	00:17:14
Camera Interaction - Physical Inspection	00:17:14	00:17:59

The grouse is inspecting the camera.

166305.375 seconds watched : 78 events marked (35 valid, 0 invalid, 0 missed)

Figure 1: An example of Wildlife@Home's video viewing interface. Users are shown 30 minute to 2 hour long nesting videos, can specify the start and end time for various events of interest, and provide tags and comments for additional detail. Users can also specify how difficult it was to determine events for the video and discuss segments of the video on the project's message boards.

quickly through the video, especially uninteresting portions, and this can cause missed events. Scientists tasked with classifying long periods of uninteresting video can tire and lose focus.

This paper investigates the use of feature detection and background subtraction for the detection of avian nesting behaviors. The feature detection in Chapter III Section 1 and Chapter IV Section 1 attempts to mimic the functionality of the human scientists by using image feature detection (SIFT [5] & SURF [6]) and a support vector machine (SVM [7–10]) to classify video frames. The background subtraction techniques in Chapter III Section 2 and Chapter IV Section 2 focus on highlighting interesting or active section of video in order to aid scientists in behavior classification.

Feature detection and machine learning are effective for detection and classification of rigid objects [11] and have also shown success in the recognition of pedestrians and some wildlife [12–16]. We tested the effectiveness of this method by using scientist observations as training data and comparing algorithm performance in recognizing bird nesting behaviors.

Background subtraction is commonly used in surveillance video as a technique for segmenting objects of interest from a scene [17, 18]. By extracting segments of the collected video with an abnormal amount of foreground activity, it is possible to algorithmically present scientists with video containing classifiable events and filter out video where no events occur.

While both methods focus on reducing scientist workload, they use very different methods to do so. The feature detection method attempts to determine an event type for each video frame by learning previous classifications made by scientists. Each video frame is tagged with ongoing events and descriptors collected from that frame are used with a support vector machine to learn bird behaviors. The goal of this process is to automatically classify nesting behaviors, especially *On Nest* and *Not In Video* events. The background subtraction focuses on eliminating work for scientists by finding sections of video with bird activity or *interesting* events. Background subtraction doesn't allow for classification of events but can greatly reduce scientist workload.

Feature detection with SURF and event classification with LIBSVM [10] has shown to be a poor performer on the Wand ildlife@Home video. Many factors may cause poor performance on the footage, including video quality, brightness fluctuations, species cryptic coloration, slightly incorrect event boundaries set by scientists, and too few features for SVM training. The poor results from this research sparked a shift to study the effectiveness of background subtraction in the same domain.

Given the diversity of species and nest locations, results find that background subtraction performance is sensitive to the amount of background movement, camera brightness, and cryptic coloration in a video. Using modern background subtraction techniques, such as Running Gaussian Average (AccAvg) [17–19], and modified versions of the ViBe [20] and Pixel-Based Adaptive Segmentation (PBAS) [21] algorithms, it is possible to show a strong correlation between scientist-observed events and those calculated with background subtraction. By

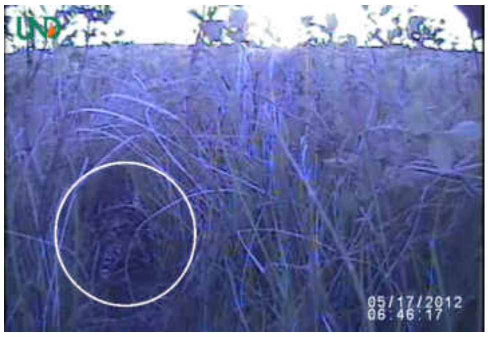
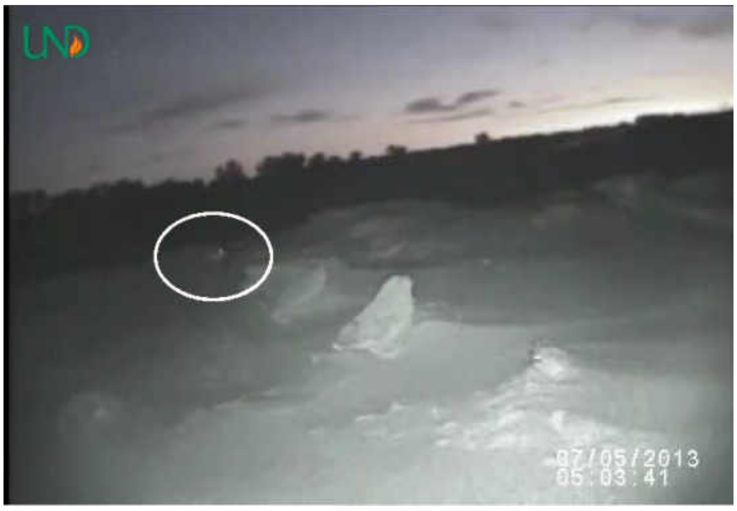


Figure 2: Sample Sharp-Tailed Grouse footage. The nest is marked with a white oval. A bird is on the nest.

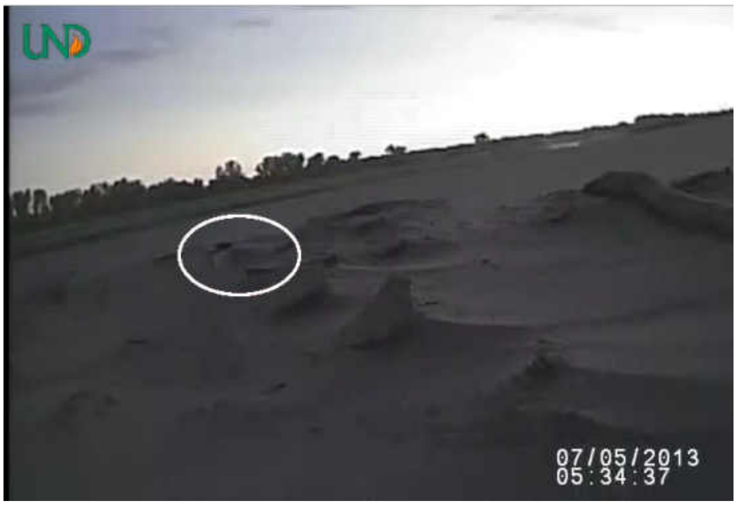
confidently narrowing the amount of video scientists are watching, it will be possible to focus on showing worth-while video segments and increase user incentive and focus.

Chapter II presents modern techniques used for common feature detection algorithms, SVMs, and background subtraction problems. Chapter III covers the approaches we took to classifying frames and extracting regions of the video with activity. Performance results and limitations of the algorithms are in described Chapter IV. Finally, Chapter V concludes with future work and a discussion of the next steps to collecting more results, improving the algorithms, and use of the data.

(a) Sample I



(b) Sample II



(c) Sample III

Figure 3: Sample sunrise Interior Least Tern footage. The nest is marked with a white oval. A bird is on the nest in all three images.

CHAPTER II
RELATED WORK

This chapter gives a detailed overview of common feature detection, machine learning methods and modern approaches to background subtraction. Both background subtraction and feature detection are popular categories of computer vision and typically play different roles in the processing of data. In the sections below we hope to provide meaningful and up to date information about applying these methods.

1 Feature Detection

The process of feature detection in computer vision is the use of image qualities to find unique or descriptive regions. These regions can be used to find a matching object or scene in another image. There are mainly three qualities of an image that are used to describe an object or scene, edges [13, 22, 23], corners [24], and blobs [5, 6, 25]. The main to algorithms we will be looking at use blob detection which uses a kernel (Laplacian or Gaussian) to find local extremum within an image and uses them as keypoints or descriptors.

1.1 SIFT: Scale Invariant Feature Detection

In this paper, Lowe [5] proposes a scale and rotation invariant feature detection and matching method called SIFT. This technique works by creating a scale space representation of the image by successively blurring the image with a Gaussian kernel. The difference of these Guassian blurs is used to locate maxima and minima in the scale space which are then used as keypoints. The image gradient $G_{i,j}$ and orientation $O_{i,j}$ are calculated using pixel differences in image I.

$$G_{i,j}(I) = \sqrt{(I_{i,j} - I_{i+1,j})^2 + (I_{i,j} - I_{i,j+1})^2} \tag{1}$$

$$O_{i,j}(I) = \arctan \frac{I_{i,j} - I_{i+1,j}}{I_{i,j+1} - I_{i,j}} \tag{2}$$

The gradient (Equation 1) and orientation (Equation 2) are stored with a canonical gradient orientation in order to make the keypoints independent of image rotation. This means storing them according to their gradient peak. Each feature is inserted into a 36 bin histogram according to their 360 degree orientation.

The scale and rotation invariance of SIFT makes it a good candidate for outdoor detection of non-rigid objects, however, the features are sensitive to lighting and this will become a problem with any nighttime footage. SIFT is also relatively slow, at around 1.5 seconds per image, and this becomes a serious problem for video processing.

1.2 SURF: Speeded Up Robust Features

Bay *e*t al. [6, 25] use a similar blob detection method to SIFT (Section 1.1). They use estimations for the Gaussian filters by taking advantage of the quick sums calculated with integral images. With an integral image, finding the sum of pixel values over any rectangular area only requires three addition operations. The use of integral images allows for extremely fast Gaussian derivatives and keypoint orientation calculations via Haar wavelets [26].

SURF comes out to be about 4 times faster than SIFT. It's speed and reliability make it better suited than SIFT for processing video, however it is still not ideal real-time video analysis.

2 Machine Learning

This section discusses the current status and role of Support Vector Machine Classification in machine learning. A Support Vector Machine (SVM) [27] is a machine learning classifier which learns how to classify new input from a set of pre-classified training data. SVMs can learn binary-class data or multiclass data.

There are two main types of optimization problems that SVMs solve in order to learn to predict the class of new data. The first equation assumes the data is cleanly separated, none of the training is incorrectly labeled. This equation is as follows:

$$\begin{aligned} D(\boldsymbol{\alpha}) = & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to } & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i, \quad i = 1, \ldots, n, \end{aligned} \tag{3}$$

such that $\boldsymbol{\alpha}$ is a set of Lagrange multipliers, $(\boldsymbol{x_i}, y_i)$ is a data element where $\boldsymbol{x_i}$ is the set of input features and $y_i \epsilon \{-1, 1\}$ is the class identifier, and n is the number of input elements.

The second equation uses a modifier called the *slack variable* which allows for a margin of error in the training data. This margin prevents overfitting of the training data. There are many different equations which use different *slack variables* but the most common is as follows:

$$\begin{aligned} D(\boldsymbol{\alpha}) = & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to } & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \end{aligned} \tag{4}$$

where the only additional symbol here is C, the *slack variable* for computing a soft margin.

Each of these equations is the dual form of their primal counterparts. This means that, under their given constraints, they return the optimal solution for their primal form.

2.1 LIBSVM: A Library for Support Vector Machines

This section is an overview of the LIBSVM software package and its implementation. LIBSVM is one of the most popular and widely used SVM libraries[10]. Chang et al. [10] emphasize LIBSVM's reliability and thus the main reason for its support and popularity in the software community. The library supports classification, regression, and distribution estimation with different SVM Kernels including, linear, polynomial, radial basis, and sigmoid.

The LIBSVM solves the C-SVM optimization problem by default. This problem is defined as:

$$\begin{aligned} \min \alpha \quad & \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q} \boldsymbol{\alpha} - \boldsymbol{e}^T \boldsymbol{\alpha} \\ \text{subject to } & \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l, \end{aligned} \tag{5}$$

where α is the set of Lagrange multipliers to be optimized, $e = [1, \dots, 1]^T$ is a vector of ones, Q is a matrix $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and where $K(x_i, x_j)$ is the Kernel function. Within Q, x is the set of features and y is the sample class. Once optimized the Lagrange multipliers are used in the classification of any test data.

$$\text{accuracy} = \frac{\text{samples correctly predicted}}{\text{total sample size}} * 100 \tag{6}$$

Chang et al. [10] measure the accuracy of an SVM classifier as the number of correctly predicated samples divided by the total sample size of the test data. Due to the reliability of the LIBSVM library the reported accuracy can be used to test future implementations of SVMs.

3 Background Subtraction

This section discusses approaches for background subtraction, or as it is sometimes referred, foreground segmentation. Background subtraction is the process of removing the uninteresting or unwanted regions of a video in order to highlight the foreground or objects of interest. Many methods fit each pixel value in a frame to a background model based on probability using previously observed values. The most common methods along with more modern approaches are presented. This includes the running Gaussian average (Section 3.1), Mixture of Gaussians (Section 3.2), ViBe (Section 3.3), pixel-based adaptive segmentation (Section 3.4), and a couple techniques used in a similar problem domain (Section 3.5).

3.1 Running Gaussian Average (AccAvg)

Running Gaussian average is one of the most basic background subtraction techniques [17, 18] and has also been effective in applications with a static background such as traffic cameras [12, 28]. This technique works by storing a model of the background $\mathbf{B}_t$ and calculating the distance of each new image $\mathbf{I}_t$ from the background model. If this distance is larger than a provided threshold, τ, then the pixel at that location is marked as foreground. This threshold can be seen in Equation 7.

$$|\mathbf{I}_t - \mathbf{B}_t| < \tau \tag{7}$$

The background model can then be updated by using an exponential moving average which slowly adapts to changes:

$$\mathbf{B}_{t+1} = \alpha \cdot \mathbf{I}_t + (1 - \alpha) \cdot \mathbf{B}_t \tag{8}$$

Where α is the rate at which the model adjusts and t is the current frame index.

There are a few effective methods for cleaning the results from a simple running Gaussian average as pointed out in [17]. The first is to clean up the foreground mask with some type of filter. Both a median filter and an open/close [29] filter work well. If a pixel has been marked as foreground for too many consecutive frames it can be set in the background model to prevent long standing false detection in the event of a sudden lighting change. Finally if a pixel is rapidly changing from foreground to background it can be masked to prevent sporadic and unreliable detection.

3.2 Mixture of Gaussians (MOG)

MOG is a widely used and robust background subtraction algorithm used in OpenCV [30]. It is based on modeling the background pixels as a combination of surfaces [31] which is further described as a Gaussian mixture model. The probability of a pixel belonging to the background is described as a sum of Gaussians:

$$f_{\mathbf{X}}(X|\mathbf{\Phi}) = \sum_{k=1}^{K} P(k) \cdot f_{\mathbf{X}|k}(X|k, \theta_k) \tag{9}$$

Where $P(k)$ is the probability of the surface k appearing in the pixel view and $f_{\mathbf{X}|k}(X|k, \theta_k)$ is the Gaussian distribution for surface k with $\mathbf{\Phi}$ being the set of theta input parameters ($\theta_k = \mu_k, \sigma_k$) for the Gaussian distributions describing each surface.

Power and Schoonees note that $P(k)$, μ_k, and θ_k are typically estimated with running averages calculated at each frame [31]. Also, $f_{\mathbf{X}|k}(X|k, \theta_k)$ for a pixel value x can be estimated by a Boolean value, true if x is within 2.5 standard deviations of the mean, false otherwise.

With MOG, similar techniques to those in Section 3.1 can be used to clean results. The use of an open/close filter is especially useful for removing noise.

3.3 ViBe

ViBe [20] is a background subtraction algorithm based on random substitution and spatial diffusion. Van Droogenbroeck *e*t al. [20] approach background model formulation with stochasticity in order to increase the robustness of their algorithms and increase the range of background pixels stored in the model. Since ViBe does not rely on statistical modeling of pixel history, the authors believe it can better match a pixel's true history by actually using past pixel values. This means ViBe can fit multimodal pixel histories and better adapt to slight background movement.

To model the background, ViBe stochastically stores 20 previous pixel values and compares new pixel values to this pixel history. If a pixel value matches (see Equation 7) two of the stored values then it is classified as part of the background, otherwise it is masked as foreground. This method of classification allows for up to 10 different background models to be fit by ViBe.

As alluded to earlier, updating the background model is a stochastic processing in ViBe. Each new observed pixel value has a 1/16 chance to overwrite a random position in the 20 previously stored pixel values. Previous pixel values are not stored as a FIFO queue since this implies some linearity to background pixel occurrence which is typically not the case in real world data. If a pixel history is updated there is another 1/16 chance to update one randomly selected neighboring pixel. This random update process allows for an adaptive model that can slowly absorb foreground object that have become part of the static background.

ViBe employs the use of an open/close filter to remove noise from the foreground mask as in 3.1. Van Droogenbroeck *e*t al. [20] also suggest using the filtered mask as the update mask such that ViBe will add the unwanted noise to the background model.

3.4 Pixel-Based Adaptive Segmentation (PBAS)

PBAS, introduced by Hofmann *e*t al. [21], is a foreground segmentation algorithm that uses the stochastic portions of ViBe [20] along with pixel-based adaptive thresholding and updating. PBAS adjusts thresholds to the pixel variance in the image by dynamically setting the threshold, τ, as shown in Equation 7, and the probability of pixel update from Section 3.3.

Hofmann *e*t al. [21] measure background dynamics by calculating the mean from a stored array of previously observed minimum pixel differences [21]. When background dynamics are high, a larger threshold, τ, can be used to reduce noise and the probability for updating the background model can be increased to allow for quicker absorption of false foreground detection. By contrast, when background dynamics are low, a smaller and more precise τ can be used with a smaller update probability to keep foreground detections in the foreground longer. This means PBAS allows for strong foreground segmentation on pixels with a highly static background while simultaneously using a more lenient set of parameters on highly dynamic regions of the image such as water or foliage.

3.5 Background Subtraction on Distributions

Work in a similar domain, the observation of avian behaviors, has been done by researching background subtraction techniques as a method for observing birds visiting a feeder [32, 33]. This environment naturally has an active background with foliage movement, however birds drawn to feeders are not typically in their ideal environment for camouflage and since they are feeding tend to be more active than when on the nest. The technique proposed in [32] was designed to solve noise generated by background movement by looking at pixel neighborhood distributions but is more computationally expensive than pixel-based approaches.

3.6 MotionMeerkat

MotionMeerkat is a general use tool to detect motion in ecological environments created by Ben Weinstein [34]. The tool is used to alleviate the process of video stream data analysis by extracting frames with motion from a video file. MotionMeerkat can either use MOG (Section 3.2) or a version of AccAvg (Section 3.1) for foreground segmentation and then uses blob detection and thresholding to determine if a foreground object it present. Weinstein's results show that MotionMeerkat is successful in many ecological environments but is still subject to problems such as rapid lighting changes, and camouflage.

CHAPTER III
METHODOLOGY

In this chapter we discuss the methods and techniques used for both the machine learning and background subtraction approaches. In Section 1 we cover the data collection process from both expert scientists (Section 1.1) and volunteer computing (Section 1.2), SVM training (Section 1.3), analysis (Section 1.4), and testing (Section 1.5). Section 2 covers the changes made to the ViBe and PBAS algorithms (Section 2.1) and the process used for converting the detected foreground into computed video events (Section 2.2).

1 Feature Detection

The feature detection research aims to automate the process of classifying the Wildlife@Home video events. Specifically the *On Nest* and *Not In Video* events. The process of collecting data, training a SVM on the data, and then testing the SVM for accuracy requires many different steps and precautions in order to maintain data accuracy. At each step a layer of complexity is added that typically requires a translation of the data into a new data type or format that can then be handled by the next process in the workflow.

The workflow uses human observations, volunteer computers, and local computers in order to finally train and test a SVM. First, a scientist must view the video and mark events that occur with a start time and an end time. The marked events are then sent to volunteer computers and the SURF algorithm is used to collect feature descriptors which can be tagged with the event types marked by the scientists. Finally these marked descriptors are used to train and test the SVM on a local machine.

1.1 Expert Classification

To understand the difficulty of the problem and for the best chance of a working classifier, we start with the most accurate data for training the SVM. This means using only video classified by experts for training as there may be errors in the volunteer classifications. Experts include anyone approved to authoritatively decide the correctness of events in a video, specifically the wildlife biologists working on the Wildlife@Home team. The classification process involves tagging events in the video along with a start time and end time for each event. Events include a variety of behaviors such as *Eggs Hatching*, *Chick Presence*, *Parent Feeding*, *Brooding*, *Nest Exchange*, *On Nest*, *Not In Video*, and many others. An example of the interface used to enter these events can be seen in Figure 1.

1.2 Descriptor Collection

Feature collection is the process of extracting features from each frame of the video using a feature extracting algorithm such as SIFT[5], SURF[6], FAST[35], HOG[13], etc. We use SURF for Wildlife@Home due to its ability to identify partially hidden objects such as the Sharp-Tailed Grouse in large amounts of foliage. SURF is sensitive to its input parameters and is tested with the input video to produce a reasonable number of features. Each feature is then converted into its location and orientation independent counterpart called a descriptor. In the case of SURF this is an array of 64 floating point values between -1 and 1. Once collected, the descriptors are added to a global array of descriptors.

This process is done for each active event type in the current frame. For example, each event in a video will have a type, such as *On Nest*, *Brooding*, *Nest Exchange*, etc. Each event contains a start time and end time. If the current frame is within the start and end time of a *Brooding* event then those descriptors will be added to the *Brooding* event type descriptor list. Likewise for overlapping events, if the frame contains both *Chick Presence* and *On Nest* then that frame's descriptors will be added to both event type descriptor lists.

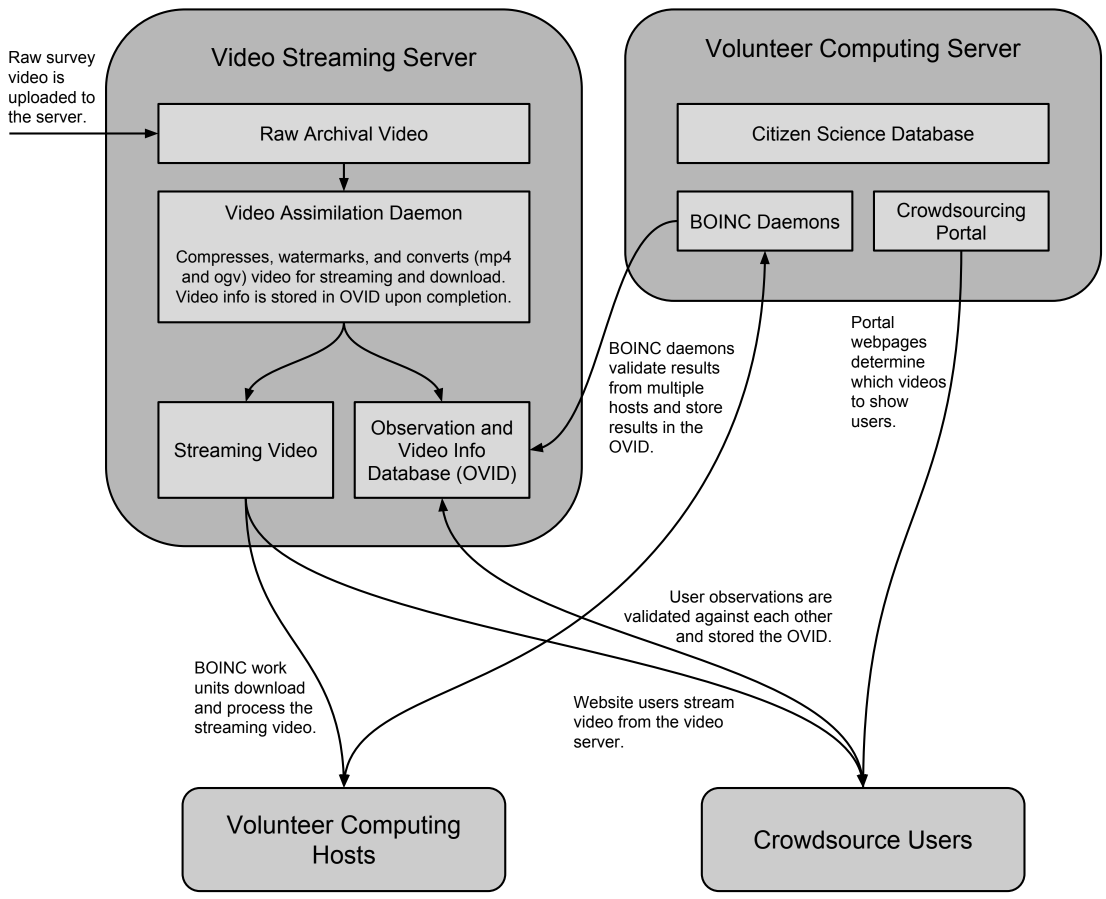

Raw survey video is uploaded to the server.
Video Streaming Server
Raw Archival Video
Video Assimilation Daemon
Compresses, watermarks, and converts (mp4 and ogv) video for streaming and download. Video info is stored in OVID upon completion.
Streaming Video
Observation and Video Info Database (OVID)
Volunteer Computing Server
Citizen Science Database
BOINC Daemons
Crowdsourcing Portal
BOINC daemons validate results from multiple hosts and store results in the OVID.
Portal webpages determine which videos to show users.
User observations are validated against each other and stored the OVID.
BOINC work units download and process the streaming video.
Website users stream video from the video server.
Volunteer Computing Hosts
Crowdsource Users

Figure 4: Wildlife@Home uses two servers to manage videos, crowdsourcing, and volunteer computing data. The video streaming server stores videos, prepares them for viewing/download and stores crowdsourcing and volunteer computing results in the OVID. The volunteer computing server houses the crowdsourcing webpages, user and host information, along with the daemons for generating and validating work.

In order to prevent the collection of duplicate descriptors between frames we match each of the existing descriptors with their nearest match in the new set using brute-force matching. We then calculate the standard deviation of these distances and only accept the new features classified as outliers.

Depending on the algorithm, parameters, video, and processor, the collection can take a few hours for each video. In our data set each video is anywhere from 30 minutes to two hours in length. In order to address the computational expense of this, we use volunteer computing with the Berkeley Open Infrastructure for Network Computing (BOINC) [36] to process each video on a volunteer host and return the collected descriptors. Each program and set of data files sent to a volunteer is called a *work unit*. The Wildlife@Home network architecture can be seen in Figure 4.

Once each work unit is completed its output is validated against a second work unit result to ensure data integrity. This is is done by validating the events and descriptors returned from each volunteer. First, the validation daemon makes sure both return the same event types, then it checks the number of descriptors returned for each type, and finally that each descriptor is a match to the descriptor from the other work unit. If there is an error at one of the steps a new work unit is sent out until a quorum is reached.

Once validated, the results are assimilated into a file structure sorted by a work unit tag, species, nest location, and video id. Each video id folder contains a file for each event type with its descriptors. These collected files can be combined or organized based on how the data needs to be analyzed.

1.3 Train SVM

For this classification problem we have chosen to use a SVM because of its ability to work with extremely complex boundaries, not only in high dimensionality but also data overlap. In addition to allowing a soft margin, SVMs also allow for a weighted margin which will train to heavily favor the correctness of one class. These SVM parameters help when training on descriptors from video where we have a lot of overlap between the positive and negative data sets.

Using the collected descriptors to train an SVM means organizing the data into two groups, a set of positive examples and a set of negative examples. This can be done in a couple different ways, however, if we want to use cross validation to test the SVM, we must partition the video files prior to combining the descriptors. We use leave-one-out validation so we choose a video which contains both the positive and negative event types to validate against.

First, we need to pick the event type or types we want to train the SVM to detect. We can either choose all non-positive events to be considered negative or we can specify the negative event types and ignore the rest. To detected bird presence we picked *On Nest* for the positive type and *Not In Video* as the negative type. These two types will minimize the overlap of positive and negative descriptors.

Next, we have the problem of the *On Nest* descriptors containing many of the *Not In Video* descriptors. This can be alleviated by finding the matches between the two sets and removing very close matches from the *On Nest* event type descriptors. This process can be ignored because a well parameterized SVM should be able to ignore the overlap, but it will make training much faster by reducing the training set size.

Once we have our training data we can begin training the SVM. For this we used LIBSVM[10]. For a general idea of training parameters we used the LIBSVM grid search program and from there customized the parameters. With our data, best results were achieved using C-SVC SVM and a Gaussian kernel:

$$e^{-\gamma|u-v|^2} \tag{10}$$

Where γ is the Gaussian kernel multiplier. Other parameters to the SVM include c and $\boldsymbol{w}$ where c is the SVM cost multiplier, and $\boldsymbol{w}$ is a vector of cost multipliers for each classification category such that w_i is a multiplier for some class i. In this case we heavily penalize all false positives and relax false negatives. This gives a heavily skewed SVM to correctly classify all negative examples. We want this skewed SVM to help detect event presence, too many false positives will invalidate the classification process. Also note that, depending on the event type and feature set, a very different set of SVM training parameters may work better for a different feature set. Results from this are in in Chapter IV Section 1.

1.4 Measure Error

Since we are using a leave-one-out cross validation technique we can get a basic understanding of how well we have trained our SVM by checking the training error. We do this by comparing the accuracy of the training examples and the testing examples. We use the following formula for accuracy:

$$A = \frac{\text{true positives} + \text{true negatives}}{\text{number of examples}} \tag{11}$$

Both the training and testing accuracy should be similar to each other but larger than the minimum accuracy shown below:

$$\min A = \frac{\max\left(\text{positives}, \text{negatives}\right)}{\text{number of examples}} \tag{12}$$

If our accuracy, A, for either the training or testing examples is equal to the minimum accuracy there is a good chance the SVM classified all examples into a single class. A quick check of the output data can confirm this. We can get a reasonably accurate SVM if A_{test} and A_{train} are both greater than the output from Equation 12 and $A_{test} \approx A_{train}$.

SVM accuracy is sensitive to the input data and parameters so finding good training data is important to establishing a reliable SVM for the classification problem.

1.5 Testing

Even with a well-trained SVM and acceptable training error, it is difficult to show that the SVM correctly classifies the descriptors from a video. In order to help test the SVM we need to somehow calculate or show that the points classified are accurate in depicting the object of interest. We do this by color coordinating the keypoints from a video by their classification. Positively classified points colored green and negatively classified points colored red. We also color points that closely match the training descriptors as blue in order to determine the accuracy of our training data. Since each frame has a very sporadic number of points accepted depending on the position of the bird or lighting we show the positively classified points (green) and training descriptors (blue) permanently. An example of this is in Figure 5.

2 Background Subtraction

The goal of this research is to determine which algorithms can best highlight regions of uncontrolled outdoor video with *interesting* events. This ideally can act as a filter and help scientists focus on segments of video that require their attention and letting them skip less interesting segments of video. The background subtraction methods need to be resistant to noise and handle quick correction of camera lighting problems while still being sensitive enough to detect the motion of a small to medium sized animal with cryptic coloration. The usefulness of these algorithms is sensitive to the number false positives and false negatives. Too many false positives and there many not be a significant length of video that can be classified as *uninteresting*, while too many false negatives may leave many *interesting* events unclassified and unwatched. An example of this is observed when comparing scientists' observations to positive events from the algorithms, as in Figure 6. An almost continuous stream of false positives can occur when vegetation moves in the wind when the grouse is not even at the nest (see Figure 6a), but on less windy days we see increased agreement between the two classifications (see Figure 6b).

Three different algorithms were evaluated for their ability to accurately detect motion in Wildlife@Home's Interior Least Tern, Piping Plover and Sharp-Tailed Grouse video. Running Gaussian Average (see Chapter II Section 3.1) was chosen as the baseline for performance as it is considered a good performer, easy to implement, and proven useful in real world applications [12, 17, 18, 28].

2.1 Algorithm Modifications

Modified versions of ViBe [20] and PBAS [21] were implemented and compared to MOG and AccAvg. ViBe is a good fit for this problem space as it is non-parametric and can be quickly initialized to prevent a large number of initial false positives. PBAS is an algorithm that adjusts its thresholding and update parameters on a pixel-by-pixel basis. PBAS is also good for this problem, where certain parts of the image are very noisy and at times entire sections of the video are polluted with dynamic lighting changes. PBAS will dynamically increase the foreground classification threshold during portions of a video affected by lighting changes and can learn to ignore regions of a video with large background variance such as in the grouse video (see Figure 2) where grass movement will span a large area of the video (100's of pixels) and pixel neighborhoods are not enough to detect the movement.

Modifications were made to improve performance on the noisy video and subjects with cryptic coloration. Initialization of ViBe and PBAS were adjusted to be second-frame-ready by adding the minimum number of values to the background model to match the first frame and filling the rest of the background model with values from random locations in the frame. This initialization allows for fast adaptation to subsequent frames if the background has a lot of motion while maintaining the minimum requirement to match the likely similar following frame. An open/close [29] filter was also added to reduce foreground detection noise in the output mask. The mathematical morphology removes small unconnected bits of noise while maintaining the larger connected regions. This prevents many false detections due to video compression and camera induced noise. Depending on the video resolution, filter size may be adjusted accordingly. Finally, in order to improve detection of birds, we use the convex hull of any connected foreground regions as the foreground mask. Since much of the birds are a similar color to their environment, generally only small areas are detected such as the head, tail feathers, and shadow, and much of the body can remain missing or segmented. The addition of a convex hull to connected foreground regions highlights bird movements and increases algorithm confidence. The convex hull may also be used in the future to detect extreme lighting changes since this will also emphasize large scene changes.

2.2 Event Calculation

The conversion from the foreground mask to calculated events is done with time-series analysis. An event is defined as a specified video segment marked with a start and an end time. Foreground pixel counts are taken as a series of data points, and these are smoothed by using an exponential moving average. This further reduces detection noise and sporadic peaks. Once the data is smoothed its mean (μ) and standard deviation (σ) are calculated and used to determine which frames have more than 3σ foreground pixels using the inequality in Equation 14. If this is the case, it is marked as a significant event, otherwise it is ignored. Experimentation can be done to determine a good threshold for the standard deviation.

The equation for the exponential moving average is:

$$m_t = \alpha \cdot x_t + (1 - \alpha) \cdot m_{t-1} \tag{13}$$

where m_t is the mean at time unit t, x_t is the number of foreground pixels at time t, and α is the weighted decrease or learning rate. As $\alpha \to 1$ the new data is more heavily weighted. An example this time-series data compared to when scientist marked events occurred can be seen in Figure 7.

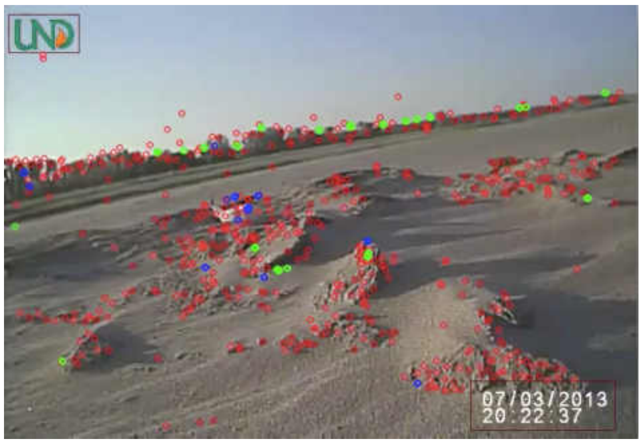
The calculation of significant foreground events is done using the following threshold inequality:

$$x_t > \mu + 3\sigma \tag{14}$$

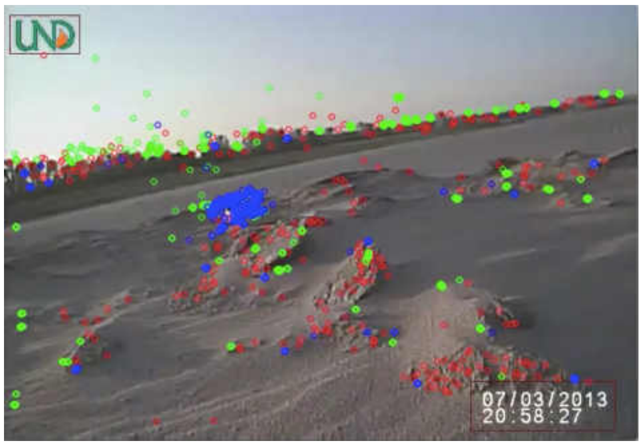
This threshold is a good indication of foreground activity even when the video has a moderate amount of noise since noisy regions are either smoothed or taken into consideration in the time-series mean. The calculated foreground activity can then be compared to scientists to determine algorithm accuracy, as shown in Figure 6. By calculating events from regions with an abnormal amount of foreground pixels, a measure for the amount of foreground activity taking place is provided. This activity can then be compared to scientists to determine the accuracy of each background subtraction algorithm as shown in Figures 6 and 7.

An example of the correlation between background subtraction events and scientist-observed events can be see in Figure 7. The arrows indicate human observed events in comparison with the time-series for each of the three algorithms. The data in these two examples are highly correlated with little noise and few false detections. It can also be observed that PBAS is very quick to adapt to changes while ViBe has the largest detection emphasis among the three algorithms.

(a) Sample I

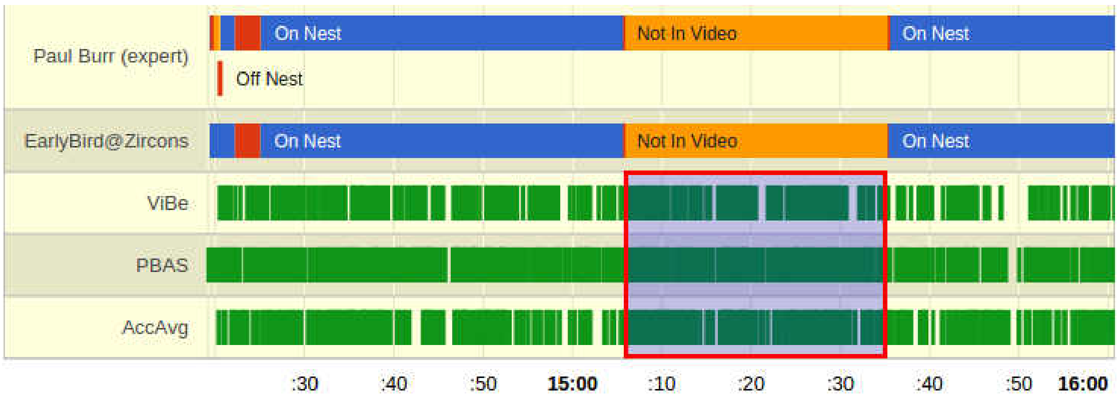


(b) Sample II

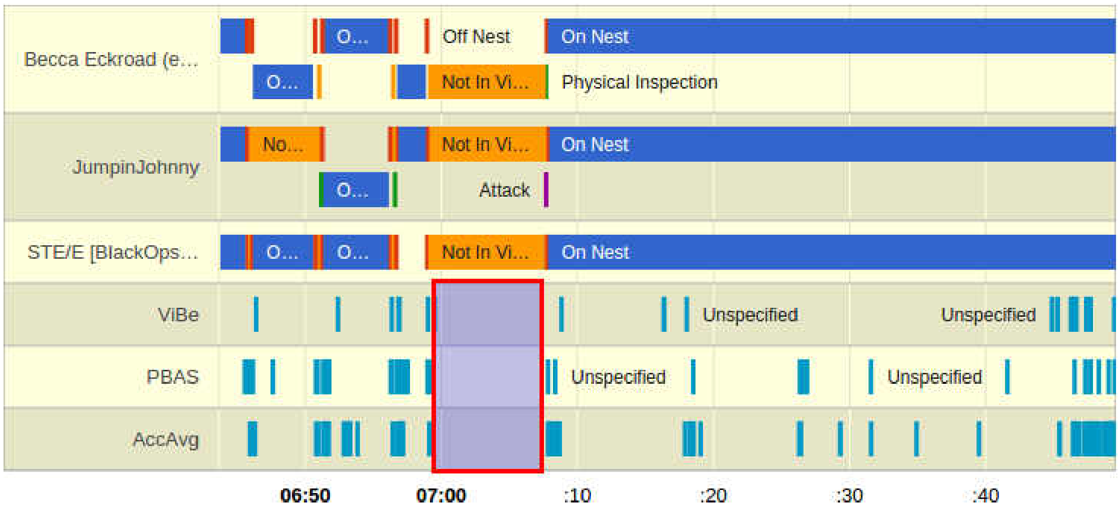


(c) Sample III

Figure 5: These samples show the progression of feature collection and classification on a Tern nest. We see a frame with no feature overlay (5a), a frame with a few features overlaid at the beginning of the video (5b), and a frame with all the features from the video overlaid (5c). Feature descriptors used for training are shown as blue circles (cumulative), additional positively classified descriptors are shown as green green circles (cumulative), and red circles indicate negatively classified descriptors (non-cumulative). Red boxes around the UND logo and the video timestamp indicate regions of the video excluded from the feature detection algorithm.

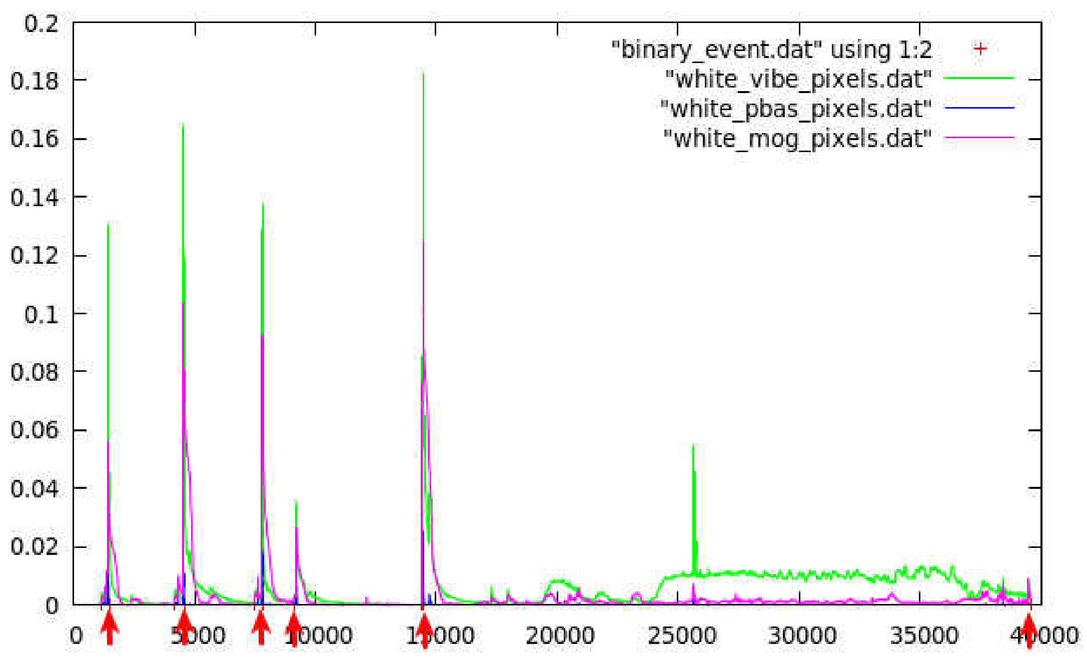


(a) Timeline I

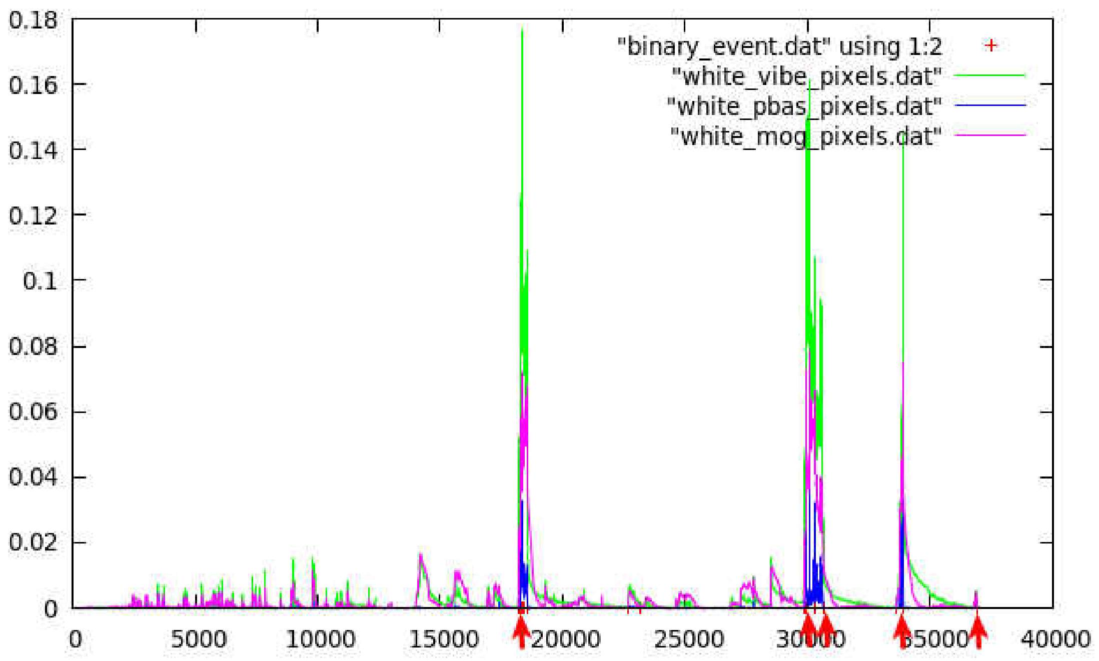


(b) Timeline II

Figure 6: Timelines showing the number of false positives in a windy grouse video (6a) against those in a less windy grouse video (6b). The highlighted regions show time segments from the background subtraction results where there is no bird on the nest. These timelines were created using the Google Charts API [37] and are easily embedded in the Wildlife@Home user interface.



(a) Sample I



(b) Sample II

Figure 7: Example of event and foreground pixel count correlation. Red arrows indicate a scientist-observed event and lines indicate foreground pixel count for each algorithm.

CHAPTER IV
RESULTS

In this chapter we cover results from both the feature detection (Section 1) and Background Subtraction (Section 2) methods. Results in each section include a summary tools used to collect data, the size of the analyzed data, and a discussion on the effectiveness of each technique.

1 Feature Detection

Approximately 500 computers processed more than 63 million images for the Wildlife@Home SVM classification project. More than 3,500 work units have been successfully processed by volunteers with Linux, OSX, and Windows computers and have collected SURF descriptors for more than 1,750 hours of wildlife video. Most work units process a single hour long video recorded at 10 frames per second. Each work unit typically takes 3 hours of CPU time depending on video length, the number of descriptors being handled, and processor speed. Depending on SURF parameters and video content each work unit returns roughly 2,000 descriptors for each event type in the video. These descriptors are stored for SVM training and testing.

For testing results, a variety of parameters were chosen for SVM training with LIBSVM. Initial parameters were determined by the LIBSVM grid search program, *grid.py*. All tested results fall in the ranges below:

$$\begin{aligned} \gamma &= 0.5 \text{ to } 10.0 \\ c &= 0.5 \text{ to } 10.0 \\ w_{-1} &= 0.5 \text{ to } 50.0 \\ w_{+1} &= 0.5 \text{ to } 50.0 \end{aligned}$$

Descriptors were collected for Sharp-Tailed Grouse, Interior Least Tern, and Piping Plover. For each species 5 different SURF minimum Hessian values were sampled, 100, 150, 200, 300, and 500. The Hessian value controls the threshold for a Hessian corner detection algorithm used in SURF for determining which points in the image to use a possible point of interest. The lower the minimum Hessian threshold value the larger the number of descriptors collected from each frame.

1.1 Video Identification Results

We selectively chose a single tern nest for testing as there is less background noise and fewer descriptors collected from the video background. Since many videos have dramatic lighting changes and indistinguishable objects at night we removed nighttime footage and videos with dramatic shadow changes from sun orientation. These videos greatly skew results from the feature detection algorithms which focus on edge and corner detection. Upon removal of these videos from our selected tern nest we have 24 acceptable videos remaining for SVM training. These videos contain 25,000 positive and 23,000 negative descriptors. Three methods could be used to provide data to the SVM as see in the list below:

1. Train on all 47,000 of the descriptors.
2. Subtract the negative features from the positive features and accept only those outside of a threshold for training.
3. Manually select the nest location with a bounding box and use the descriptors in the box as positives.

Method 2 reduces the positive set size from 47,000 to 120 descriptors. Method 3 results in a positive set of about 8000 descriptors. Method 1 is the slowest and hardest to train because of the volume of features along with the large overlap in positive and negative features. Method 2 is easiest to train and theoretically the most accurate, however the difficulty comes in choosing the descriptor subtraction threshold. Too large of threshold leads to training on video outliers such as video artifacts and too large a threshold will continue to have similar problems as method 1. Method 3 can work well but requires manual selection of the nest location and will not work if the bird is not always directly on the nest. We used method 3 in our system to reduce the system memory footprint and SVM training time, reduce the chance of a possible event classification error, and to test the algorithms in the best possible scenario. A classification could happen when an expert marks *Not In Video* when the bird has just appeared or possibly quickly entered and exited the camera view. This type of error will cause SURF to misclassify the extracted features from the incorrectly marked frames and feed incorrect data to the SVM.

Results for this sample data using method 3 can be seen in figures 5b and 5c. Keypoints from the video were drawn on each frame and colored according to their SVM classification. The negative SVM classifications are colored red, positive classifications colored green, and the closest matches to training data are colored blue. The green and blue points were redrawn on all successive frames to show a clustering and get an idea of their overall representation. As seen in Figure 5b there is very little correlation in the position of points and the location of the bird but as the video progresses more and more training points start to position themselves on the bird and around the nest in Figure 5c. However, since the majority of green points aren't necessarily enclosed in the same region it is likely the SVM is not being optimally trained on the descriptors. This is a sign of overfitting, specifically on any video artifacts that are appearing only in the positively correlated frames such as *parent on nest*. If we saw a larger number of positive classifications than negative classifications it would be a sign of underfitting the data and accepting too many descriptors.

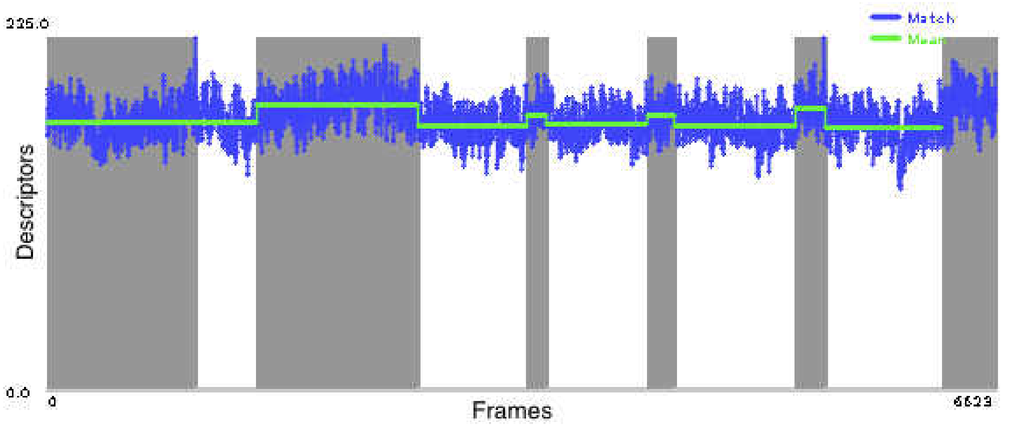


Figure 8: Single tern video descriptors run against itself with training features using method 3 for SVM training. White segments indicate frames where an expert has marked *On Nest* and gray regions are segments where the expert marked *Not In Video*. The blue line indicates the number of descriptors found to match a *On Nest* event. The green line represents the average number of matched descriptors for its corresponding gray or white time segment.

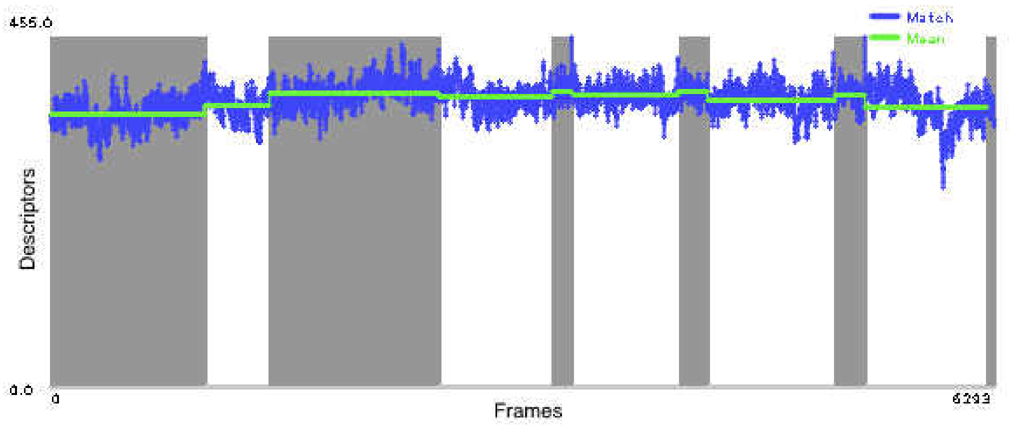


Figure 9: Three tern videos using leave-one-out cross validation and method 3 for SVM training. White segments indicate frames where an expert has marked *On Nest* and gray regions are segments where the expert marked *Not In Video*. The blue line indicates the number of descriptors found to match a *On Nest* event. The green line represents the average number of matched descriptors for its corresponding gray or white time segment.

1.2 Tern Presence Correlation

In order to predict bird presence we selected three Interior Least Tern videos from the same nest and charted the number of descriptor matches against the expert classified bird presence events for that video. Results can be seen in Figure 8 and 9 where the gray background indicates bird presence, white background indicates bird absence, the blue line is the number of matching descriptors in each frame, and the green line is the mean for that segment of either bird presence or absence. Figure 8 shows results for a single video's descriptors tested against itself and Figure 9 shows results using descriptors from videos at the same nesting site against the same video used in Figure 8. As seen in each case there is a signal indicating an increase in matching descriptors when the bird is present and a decrease when the bird is absent. The optimal case in Figure 8 has only a slightly more pronounced signal which indicates descriptors across videos from the same nest have similar values.

1.3 Effectiveness of Feature Detection

Results from this feature detection research shows that there may be a slight correlation in the number of SVM matches and bird presence in a video. This however is not a good indication since we have only trained and tested on the best possible scenarios and still see mixed results (Figures 8 and 9).

Many factors may be causing the poor performance from the feature detection. Video quality and resolution, random noise detected during intra coded frames from video compression, moving shadows from outdoor video, and a large number of overlapping descriptors from frames with a bird and frames without a bird can all be causes of poor SVM performance. It is likely that many of these factors contribute to the results we see in Figures 8 and 9.

2 Background Subtraction

Three background subtraction algorithms — AccAvg, ViBe, and PBAS — were run against a set of nearly 70,000 tern and plover videos and more than 22,000 grouse videos. MOG was not used with this data because the OpenCV [30] implementation at time time of this writing hides MOG's internal states and is not serializable for checkpointing on BOINC clients. The plover and tern video totals approximately 3 years and 9 months of footage, and the grouse video totals more than 2 years. Videos range anywhere from 30 minutes to 2 hours in length. Each algorithm runs at more than 10 frames per second (the recording frame rate) on a hyperthreaded 3.5 GHz core processor and is considered capable of real-time processing. Results were collected using BOINC and about 300 volunteers. Processing took only 10 days to run the more than 31 years of CPU time required. Results were compared to more than 20 weeks worth of observations made by project expert scientists and volunteer citizen scientists to determine algorithm accuracy.

2.1 Detecting Events with Background Subtraction

Tables 1, 2, 3, and 4 present how well each algorithm matched up to project scientists and volunteers for Sharp-Tailed Grouse, and Piping Plover and Interior Least Tern combined. Piping plover and least tern results were combined as the birds and environments are similar, and both species are being observed for the same set of events. The *Event Count* column shows the total number of each event that occurred in the set of videos analyzed, and the following columns present how many of those events the background subtraction algorithm found.

Any background subtraction detected events that occur within 30 seconds of the start or end time of a scientist-observed event are marked as a match. Multiple matches to the same start and end event from the same scientist are ignored. Since all three algorithms are adaptive, learning takes place in each algorithm where it will begin to ignore bird presence and absence on the nest.

Table 1: Algorithm Accuracy vs Expert Scientists on Tern and Plover Nests

Event Type	Event Count	AccAvg	ViBe	PBAS
Preen	274	253	123	187
Scratch	30	30	7	10
Shake	10	10	0	6
Not In Video	1684	1402	769	1432
Brooding Chicks	2	1	1	1
Nest Exchange	46	36	26	38
Foraging	228	212	54	92
Adult-to-Adult Feed	32	20	16	20
Human	8	4	4	6
Nest Defense	12	12	4	10
Predator	12	4	3	8
Non-Predator Animal	22	16	9	14
Unspecified	400	80	48	90
On Nest	2284	1753	847	1621
Off Nest	3158	2548	1582	2489

Table 2: Algorithm Accuracy vs Citizen Scientists on Tern and Plover Nests

Event Type	Event Count	AccAvg	ViBe	PBAS
Not In Video	722	630	331	682
Brooding Chicks	42	40	13	42
Nest Exchange	140	128	71	127
Eggshell Removal	2	2	1	2
Foraging	130	127	60	117
Adult-to-Adult Feed	22	22	18	22
Human	8	8	8	8
In Video	60	34	9	52
Eggs Hatching	2	0	0	2
Foraging	14	14	5	14
Adult-to-Chick Feed	36	30	25	36
Nest Defense	6	6	4	6
Predator	2	2	0	2
Non-Predator Animal	68	55	32	62
Unspecified	38	33	25	32
On Nest	1158	1059	525	1052
Too Dark	4	0	1	1
Off Nest	1454	1339	699	1434

Table 3: Algorithm Accuracy vs Expert Scientists on Grouse Nests

Event Type	Event Count	AccAvg	ViBe	PBAS
Not In Video	988	849	473	909
Brooding Chicks	2	2	1	2
Eggshell Removal	8	8	4	8
Human	8	8	6	8
In Video	320	290	289	303
Predator	34	24	6	28
Non-Predator Animal	16	11	8	12
Unspecified	14	13	6	13
Attack	6	6	0	6
Physical Inspection	92	87	25	87
Observation	78	77	35	75
On Nest	916	721	467	739
Video Error	18	7	6	14
Camera Issue	2	1	1	2
Off Nest	1888	1692	984	1787

Table 4: Algorithm Accuracy vs Citizen Scientists on Grouse Nests

Event Type	Event Count	AccAvg	ViBe	PBAS
Not In Video	3626	3194	1658	3304
Walking	2	2	2	2
Brooding Chicks	22	20	9	19
Eggshell Removal	104	97	41	99
Human	44	44	8	40
In Video	432	371	322	370
Eggs Hatching	14	13	5	12
Nest Defense	6	6	4	6
Predator	126	118	36	121
Non-Predator Animal	74	62	35	66
Unspecified	48	40	20	45
Attack	84	84	43	83
Physical Inspection	168	164	78	168
Observation	94	91	60	93
On Nest	3812	3039	1640	3116
Too Dark	10	5	2	5
Video Error	86	20	27	58
Camera Issue	18	8	5	12
Off Nest	6296	5874	2948	6096

Event start and end times that take place within the first 10 seconds of the beginning of the videos were ignored as the algorithms did not have time to learn an initial background yet.

Tables 5, 6, 7, and 8 compare results from combining all three background subtraction algorithms. The *Any Alg* column shows the number of events that matched any one of the three algorithms, and the *All Alg* column shows the number of events that matched all three algorithms. Using events marked by any algorithm provided an increase in events detected over PBAS for all event types and using a consensus showed a decrease in the number of events found. Both of these behaviors indicate that the events detected by the algorithms are not subsets of one another. This may lead to a combination of algorithms to increase overall accuracy but will also combine the false positives from each algorithm.

2.2 Analysis of False Positives

Tables 9 and 10 provide an analysis of false positives generated by the background subtraction algorithms. False positives were counted by the number of computer classified events that occur during a user classified *Not In Video* event. Results are reported as the mean (μ) and standard deviation (σ) of false positives occurring during any *Not In Video* event by any scientist over all videos tested for that species. Videos without a *Not In Video* event were ignored to prevent padding the results. A 10 second buffer is used after the start and before the end of the *Not In Video* events to avoid counting edge case movement as a false positive. This was used as a measure for false positives since at any other time a detection may correspond to an unmarked event, such as motion from the bird on the nest.

2.3 Effectiveness of Background Subtraction

The background subtraction results in Tables 1, 2, 3, and 4 show that background subtraction is accurate enough to be a reliable detection method for

Table 5: Algorithm Accuracy with Consensus vs Expert Scientists on Tern and Plover Nests

Event Type	Event Count	Any Algorithm	All Algorithms	AccAvg & ViBe	AccAvg & PBAS	ViBe & PBAS
Preen	274	260	117	121	180	119
Scratch	30	30	7	7	10	7
Shake	10	10	0	0	6	0
Not In Video	1684	1537	695	705	1305	751
Brooding Chicks	2	1	1	1	1	1
Nest Exchange	46	40	24	24	36	24
Foraging	228	216	48	53	89	48
Adult-to-Adult Feed	32	20	16	16	20	16
Human	8	6	4	4	4	4
Nest Defense	12	12	3	4	10	3
Predator	12	11	1	1	1	3
Non-Predator Animal	22	21	7	7	9	9
Unspecified	400	96	44	44	76	46
On Nest	2284	1889	758	786	1499	805
Too Dark	2	0	0	0	0	0
Off Nest	3158	2741	1432	1462	2307	1541

Table 6: Algorithm Accuracy with Consensus vs Citizen Scientists on Tern and Plover Nests

Event Type	Event Count	Any Algorithm	All Algorithms	AccAvg & ViBe	AccAvg & PBAS	ViBe & PBAS
Not In Video	722	692	300	304	625	322
Brooding Chicks	42	42	12	12	40	13
Nest Exchange	140	130	71	71	125	71
Eggshell Removal	2	2	1	1	2	1
Foraging	130	129	55	58	116	56
Adult-to-Adult Feed	22	22	18	18	22	18
Human	8	8	8	8	8	8
In Video	60	54	7	7	32	9
Eggs Hatching	2	2	0	0	0	0
Foraging	14	14	5	5	14	5
Adult-to-Chick Feed	36	36	23	23	30	25
Nest Defense	6	6	4	4	6	4
Predator	2	2	0	0	2	0
Non-Predator Animal	68	65	28	28	52	32
Unspecified	38	38	21	21	29	23
Observation	2	0	0	0	0	0
On Nest	1158	1115	485	498	997	511
Too Dark	4	1	0	0	0	1
Off Nest	1454	1450	645	655	1327	685

Table 7: Algorithm Accuracy with Consensus vs Expert Scientists on Grouse Nests

Event Type	Event Count	Any Algorithm	All Algorithms	AccAvg & ViBe	AccAvg & PBAS	ViBe & PBAS
Not In Video	988	945	423	434	817	458
Brooding Chicks	2	2	1	1	2	1
Eggshell Removal	8	8	4	4	8	4
Human	8	8	6	6	8	6
In Video	320	312	264	269	281	284
Predator	34	28	6	6	24	6
Non-Predator Animal	16	12	7	7	11	8
Unspecified	14	13	6	6	13	6
Attack	6	6	0	0	6	0
Physical Inspection	92	91	24	24	83	25
Observation	78	78	35	35	74	35
On Nest	916	813	400	415	658	441
Video Error	18	16	3	3	5	6
Camera Issue	2	2	1	1	1	1
Off Nest	1888	1850	889	911	1635	956

Table 8: Algorithm Accuracy with Consensus vs Citizen Scientists on Grouse Nests

Event Type	Event Count	Any Algorithm	All Algorithms	AccAvg & ViBe	AccAvg & PBAS	ViBe & PBAS
Not In Video	3626	3433	1548	1579	3075	1617
Walking	2	2	2	2	2	2
Brooding Chicks	22	22	8	9	17	8
Eggshell Removal	104	104	38	38	94	39
Human	44	44	8	8	40	8
In Video	432	418	271	287	330	299
Eggs Hatching	14	13	5	5	12	5
Nest Defense	6	6	4	4	6	4
Predator	126	122	34	34	117	36
Non-Predator Animal	74	67	32	32	61	35
Unspecified	48	45	16	16	40	20
Attack	84	84	43	43	83	43
Physical Inspection	168	168	75	75	164	78
Observation	94	94	58	59	90	59
On Nest	3812	3384	1453	1501	2797	1566
Too Dark	10	5	2	2	5	2
Video Error	86	65	12	12	16	24
Camera Issue	18	12	3	3	8	5
Off Nest	6296	6236	2796	2833	5744	2901

Table 9: Algorithm False Positives vs Expert Scientists

		AccAvg		ViBe		PBAS	
Species	Observations	μ	σ	μ	σ	μ	σ
Grouse	790	173.59	241.17	123.77	227.56	203.22	338.57
Tern	99	1.78	7.85	2.04	9.70	1.54	6.78
Plover	239	2.36	10.34	1.02	2.87	1.08	3.07

Table 10: Algorithm False Positives vs Citizen Scientists

		AccAvg		ViBe		PBAS	
Species	Observations	μ	σ	μ	σ	μ	σ
Grouse	3909	263.72	292.61	180.23	237.15	262.29	305.48
Tern	58	0.41	1.77	1.26	2.41	0.93	2.49
Plover	118	11.53	32.83	5.74	14.82	4.47	8.92

all three species of video. However we see too many false positives on the windy grouse video for this method to be a useful indicator of event occurrence in those videos. For all other videos, especially in the case of the *Not In Video*, *On Nest*, and *Off Nest* events, the detection accuracy is high enough to be useful for decision making. The other event sample sizes are still too small, requiring more scientist observations. Both AccAvg and PBAS have high accuracy and a low number of false positives on the tern and plover video. We see PBAS has a slightly higher accuracy on the *Not In Video* and *On Nest* events which are the main indicator events for species transitions into and out of the frames. Interestingly we see AccAvg with a high accuracy on the presumably more difficult to detect events such as *Scratch*, *Shake*, and *Foraging* events in Table 1. PBAS is likely the best overall performing algorithm for scientist use due to its low false positive rate and high accuracy on bird presence indicator events.

The grouse have the highest average number of false positives (Tables 9 and 10) and by far the highest standard deviation of false positives. The large number of false positives is caused by the more active backgrounds (moving vegetation) of the grouse. This negatively impacts the usefulness of the grouse results. However, the high variance in the grouse results suggest that some videos may have a low number of false positives, indicating better precision on less windy videos. We can see this in Figures 6 and 10 where the algorithms all have a high number of false positives in Figures 6a and 10a and little to no false positives in Figures 6b and 10b. This also indicates that the high accuracy on the grouse videos (Tables 3 and 4) is not solely false positives due to moving foliage.

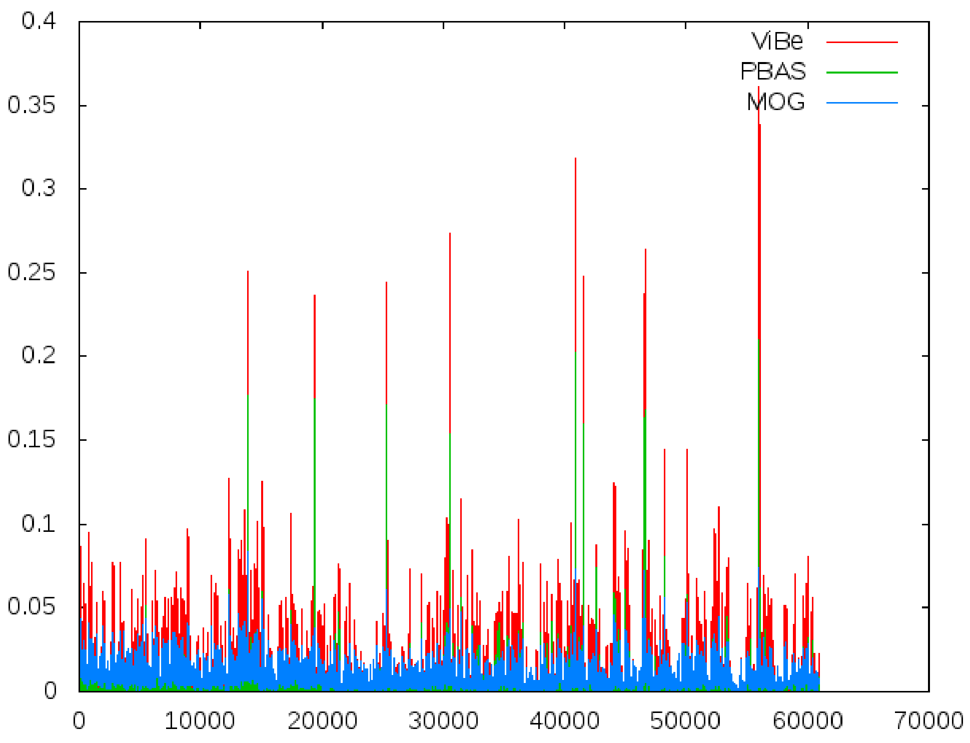
The usefulness of the background subtraction algorithms can be seen in Figures 11, 12, and 13. In these figures we compare the number of useful videos for each species by thresholding on percentage of false positives in the video. This percentage is calculated as the number of seconds of false detection relative to the number of seconds of *Not In Video* events for each video. We see that the grouse videos suffer from a far higher percentage of false positives than the tern and plover video however above a 20% threshold we see good performance for the grouse. The tern and plover video requires only a threshold of about 5% false positives.

2.4 Background Subtraction Inaccuracy and Errors

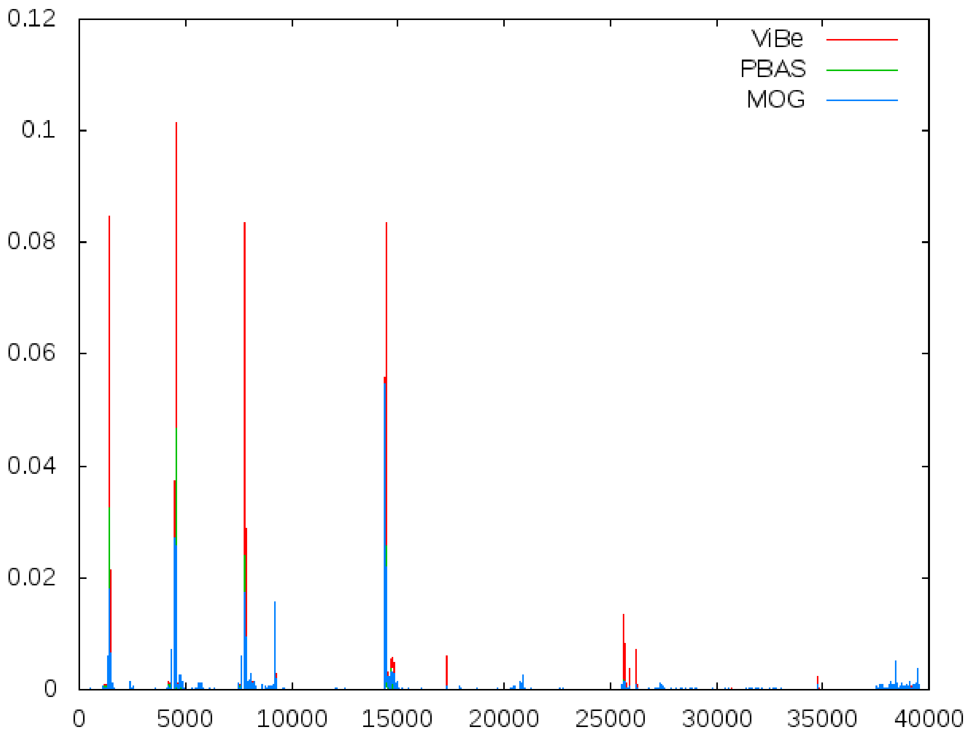
A major cause for algorithm inaccuracy and large variance in false positives (especially in the Least Tern samples) is from camera lighting autocorrection discussed in Section 2 and seen in Figure 14. Changes in scenery brightness from transitions in time of day or significant overhead cloud movement cause the camera to adjust brightness and can cause large scale false foreground detection. If the camera rapidly and repeatedly changes the brightness we see regions of video that the foreground algorithms cannot adapt to as shown in Figure 14. Due to the nature of PBAS, it adjusts to the rapid brightness changes but this still causes false negatives if a scientist-observed event does occur during or shortly after the brightness adjustments.

Other detection errors are caused by video compression noise, and species cryptic coloration. The original archival Wildlife@Home videos taken by the field cameras are compressed by the hardware in part due to storage reasons. With these background subtraction algorithms working on moderate to heavy compression, false positives are caused during transitions between intra coded frames. More sensitive events such as preens and scratches can be difficult to detect due to the small amount of motion involved (typically just body rotation and head movement) given the camera distance and resolution, along with the cryptic coloration of the species. With the surrounding area taking on such a similar color to the bird a simple preen or scratch may easily go undetected by a background subtraction algorithm.

It is also worth noting that many detected events may not line up with the start or end time of a scientist observation but may still be a cause of bird motion. For example in Figure 6b, no events occur while the bird is off the nest, but we see sporadic events while it is on the nest. This could be caused by bird adjustment on the nest or unmarked bird grooming events. The frequency of events occurring during a video may also serve as an additional indicator of bird presence, and merits further investigation.



(a) Sample I



(b) Sample II

Figure 10: Example of foreground pixel noise in a windy grouse video vs a non-windy grouse video. This figure corresponds to the timelines in Figure 6. The noise from foreground motion is very significant in the windy video. The magnitude of the events in Figure 10b would be lost in noise if the same events occured in Figure 10a.

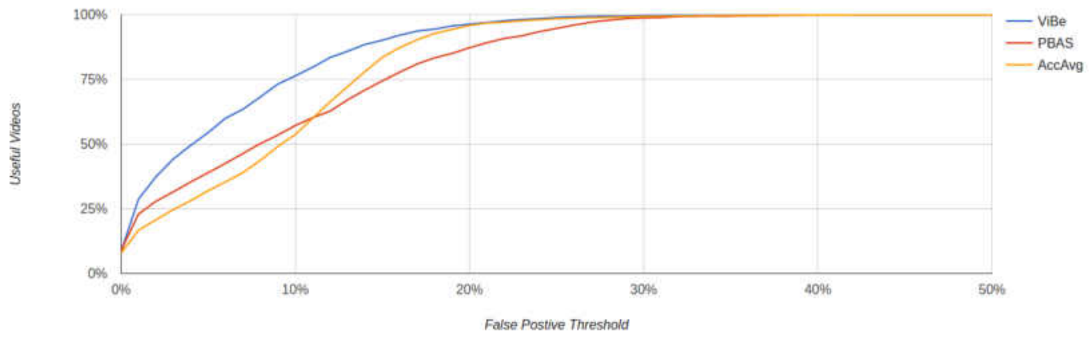


Figure 11: Percentages of grouse videos calculated as *useful* in relation to a false positive threshold.

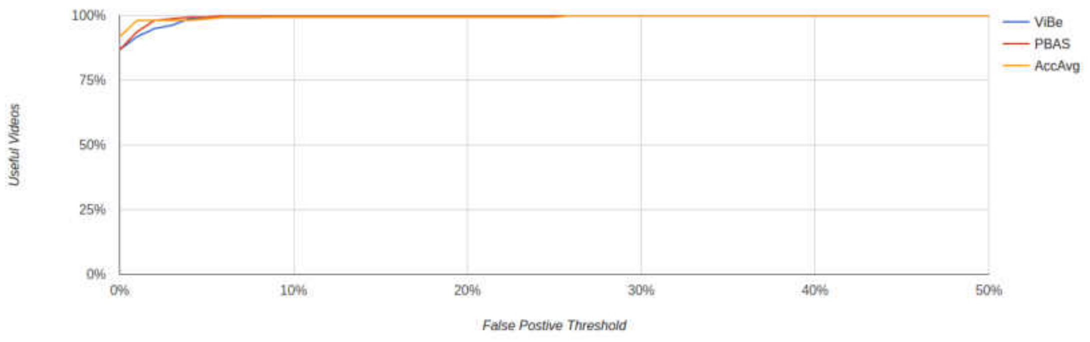


Figure 12: Percentages of tern videos calculated as *useful* in relation to a false positive threshold.

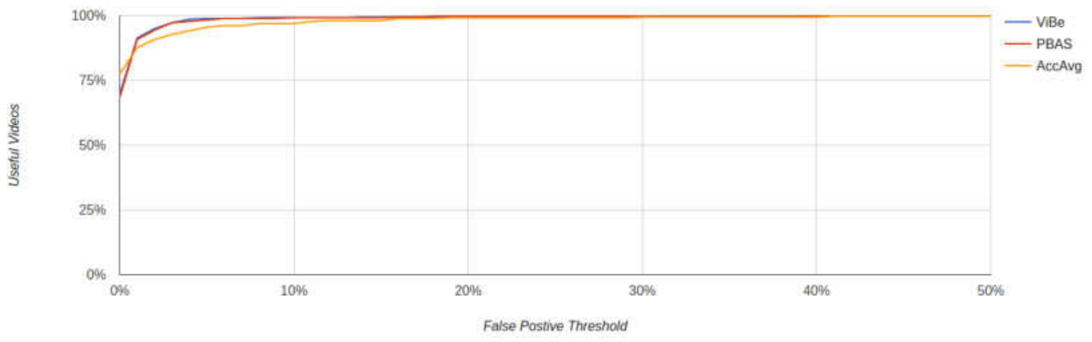
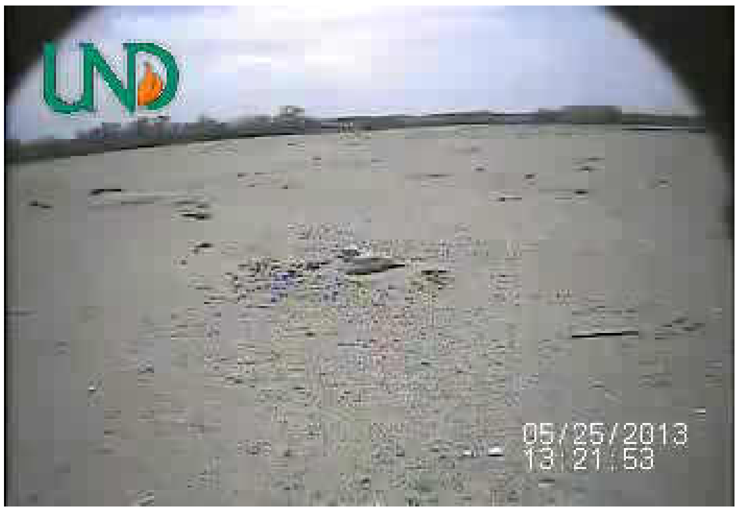
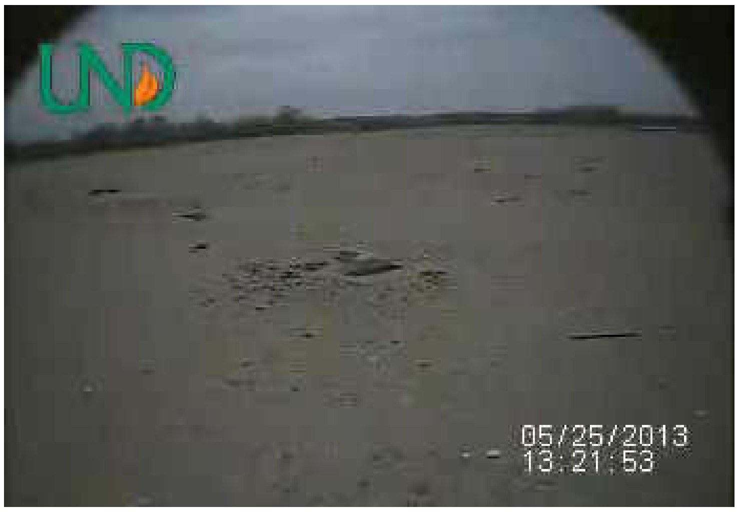


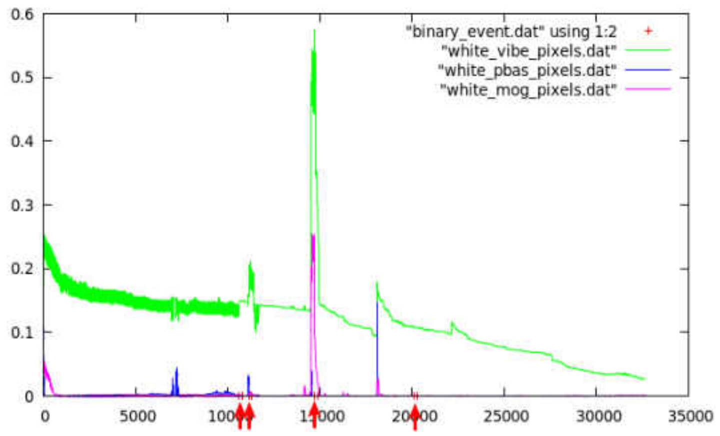
Figure 13: Percentages of plover videos calculated as *useful* in relation to a false positive threshold.



(a) Sample I



(b) Sample II



(c) Sample Issue Foreground Count

Figure 14: Rapid and repeating brightness adjust caused by overhead cloud movement. Brightness is alternated multiple times per second creating a messy foreground pixel timeline show in Figure 14c. ViBe fails to adapt to the rapid changes and both MOG and PBAS become ignorant to small pixel changes required to detect bird movement.

CHAPTER V
CONCLUSION

This paper presents an analysis of the use of feature detection and background subtraction algorithms for the classification and detection of events within uncontrolled outdoor avian nesting video. The effectiveness of feature detection was tested with SURF and a SVM while background subtraction was tested with MOG, Running Gaussian Average, modified ViBe, and modified Pixel-Based Adaptive Segmentation.

Feature detection using SURF was run using BOINC [36] and used approximately 500 machines to process 1,750 hours of wildlife video. Each machine takes approximately 3 hours of CPU time to process a single hour long video recorded at 10 frames per second. The descriptors collection from the volunteer computers were processed on a 4 core MacBook Pro using LIBSVM [10].

The results for the background subtraction algorithms where obtained using over 68,000 hours of video along with more than 20 weeks of human observations gathered by project experts and volunteer citizen scientists at the Wildlife@Home project [1, 2]. Results show that AccAvg and PBAS perform quite well and reach high enough accuracy to be promising techniques for detecting video segments that are most interesting and important for an expert and citizen scientist to observe and classify. This opens up the possibility of using the modified PBAS or AccAvg as a filter to reduce the amount of time spent by scientists analyzing the 68,000 hours of video at Wildlife@Home.

1 Feature Detection

Results for learning to detect the presence of birds in wildlife video are not promising. Too many variables come into play with video quality, cryptic coloration, camera lighting, and the quality of training data. The feature detection used requires very clean training data and this couldn't be provided by the current Wildlife@Home event classification system.

Certain aspects of this approach may see improvement with some changes. The effect of camera lighting on descriptor quality may be reduced with the normalization of the video lighting using something like Retinex [38, 39]. The quality of the training data may see some improvements with the user of a buffer on expert events. This could be implemented by ignoring frames near the beginning and end of expert events to help prevent misclassifying descriptors collected by SIFT and SURF.

Using a feature detector that can handle non-rigid objects, such as HOG [13], may reduce the number of SVM false matches. Since HOG focuses on gradient changes rather than the detection of corners it may be a better approach to feature detection.

2 Background Subtraction

Background subtraction shows promise as a useful technique for reliably detecting interesting video in the Wildlife@Home tern and plover video. The number false positives in the grouse footage makes it less useful for for scientists but it remains an accurate method of detecting movement. With PBAS and AccAvg having relatively high accuracy and low number of false positives they are currently the best overall performers.

In addition to analyzing more videos, changes can be made in order to more accurately detect segments of interest within the videos. Rapidly changing brightness inhibits the background subtraction algorithms. Possibilities for normalizing scene brightness, such as Retinex [38, 39] or adjusting the exponential moving average filter to mark video segments with extreme foreground detection (*e.g.*, larger then 20% to 30% of the frame) remain as future work. More in-depth improvements could involve taking nest location into consideration and increasing the importance of foreground pixels located around

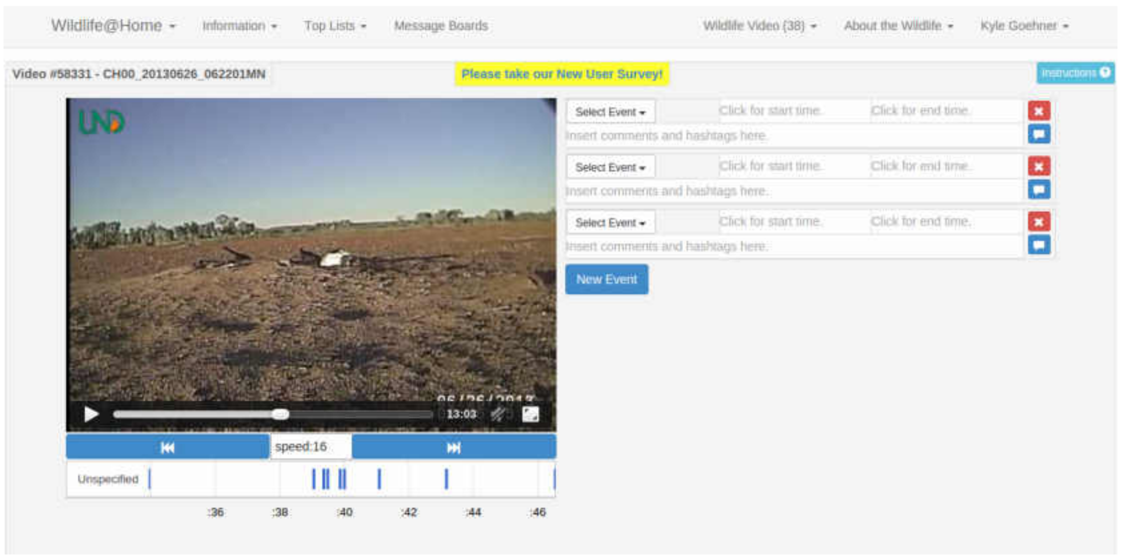


Figure 15: An example of Wildlife@Home's video viewing interface with computed events times. Computed event times can be seen in a timeline below the presented video.

the nest. Since cameras are placed strategically facing the nests we can safely assume nest location is close to the center of the frame and can easily scale foreground pixel importance accordingly.

These background subtraction results have been integrated into the video watching interface (Figure 15) used by project and citizen scientists to gain human feedback on the correctness of computer calculated event occurrences. This will not only help confirm the computed results but will also notify users to a possible upcoming event which could improve human accuracy. Background subtraction provides a first step towards using automated strategies as a filter before showing the Wildlife@Home videos to scientists and allowing them to reliably skip segments of the videos where there is no animal activity.

3 Future Work

This thesis opens up interesting questions and work for the future. Can results for windy grouse video see performance by emphasizing motion that appears towards the center of the frame? Would removing large blobs of detected foreground regions reduce the number of false detections from rapid brightness changes? How accurately we predict bird presence by the frequency of detected events? How useful are these results to scientists? How useful will scientists perceive these predictions to be? Will the use of these computed events increase the speed or accuracy of expert and citizen scientists?

4 Final Thoughts

With many scientists turning to surveillance video as a form of data collection they are limited mainly by the work required to analyze and classify the data. In the case of Wildlife@Home this is more than 68,000 hours of video. This thesis examines the used of two different methods to automatically or semi-automatically reduce the effort required to analyze video. The feature detection approach had problems classifying events with the low quality, 24/7 outdoor video. Background subtraction fared much better with event accuracy above 90% for the modified PBAS algorithm. Results show that this work is general enough to be applied to surveillance video with calm environments as a method of pre-processing to highlight segments of video with motion events.

BIBLIOGRAPHY

[1] T. Desell, R. Bergman, K. Goehner, R. Marsh, R. VanderClute, and S. Ellis-Felege, “Wildlife@Home: Combining crowd sourcing and volunteer computing to analyze avian nesting video,” in *eScience (eScience), 2013 IEEE 9th International Conference on.* IEEE, 2013, pp. 107–115.

[2] T. Desell, K. Goehner, A. Andes, R. Eckroad, and S. Ellis-Felege, “On the effectiveness of crowd sourcing avian nesting video analysis at Wildlife@Home,” in *The 15th International Conference on Computational Science*, Reykjavík, Iceland, June 2015.

[3] W. A. Cox, M. S. Pruett, T. J. Benson, J. C. Scott, and F. R. Thompson, “Development of camera technology for monitoring nests,” *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 185–210, 2012.

[4] S. N. Ellis-Felege and J. P. Carroll, “Gamebirds and nest cameras: present and future,” *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 35–44, 2012.

[5] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer Vision–ECCV 2006.* Springer, 2006, pp. 404–417.

[7] C. J. Burges, “A tutorial on support vector machines for pattern recognition,” *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[8] A. Carpenter, "Cusvm: A cuda implementation of support vector classification and regression," *patternsonscreen. net/cuSVMDesc. pdf*, 2009.

[9] E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui, "Psvm: Parallelizing support vector machines on distributed computers," in *NIPS*, 2007, software available at http://code.google.com/p/psvm.

[10] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] A. Faro, D. Giordano, and C. Spampinato, "Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1398–1412, 2011.

[12] J. Heikkilä and O. Silvén, "A real-time system for monitoring of cyclists and pedestrians," *Image and Vision Computing*, vol. 22, no. 7, pp. 563–570, 2004.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[14] X. Fan, S. Mittal, T. Prasad, S. Saurabh, and H. Shin, "Pedestrian detection and tracking using deformable part models and kalman filtering," *Journal of Communication and Computer*, vol. 10, pp. 960–966, 2013.

[15] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp. 193–199.

[16] B. J. Boom, P. X. Huang, C. Beyan, C. Spampinato, S. Palazzo, J. He, E. Beauxis-Aussalet, S.-I. Lin, H.-M. Chou, G. Nadarajan *et al.*, "Long-term underwater camera surveillance for monitoring and analysis of fish populations," *VAIB12*, 2012.

[17] A. M. McIvor, "Background subtraction techniques," *Proc. of Image and Vision Computing*, vol. 4, pp. 3099–3104, 2000.

[18] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.

[19] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 780–785, 1997.

[20] M. Van Droogenbroeck and O. Barnich, "Vibe: A disruptive method for background subtraction," *Background Modeling and Foreground Detection for Video Surveillance*, 2014.

[21] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on.* IEEE, 2012, pp. 38–43.

[22] W. Gao, X. Zhang, L. Yang, and H. Liu, "An improved sobel edge detection," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5. IEEE, 2010, pp. 67–71.

[23] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[24] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on.* IEEE, 1994, pp. 593–600.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[26] C. K. Chui, *An introduction to wavelets.* Academic press, 2014, vol. 1.

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision &amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE, 1994, pp. 126–131.

[29] G. Matheron and J. Serra, "Image analysis and mathematical morphology," 1982.

[30] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.

[31] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proceedings image and vision computing New Zealand*, vol. 2002, 2002, pp. 10–11.

[32] T. Ko, S. Soatto, and D. Estrin, "Background subtraction on distributions," in *Computer Vision–ECCV 2008.* Springer, 2008, pp. 276–289.

[33] ——, "Warping background subtraction," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 1331–1338.

[34] B. G. Weinstein, "Motionmeerkat: integrating motion video detection and ecological monitoring," *Methods in Ecology and Evolution*, 2014.

[35] E. Rosten, G. Reitmayr, and T. Drummond, "Real-time video annotations for augmented reality," in *Advances in Visual Computing.* Springer, 2005, pp. 294–302.

[36] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on.* IEEE, 2004, pp. 4–10.

[37] Google Inc. (2015) Google charts. [Online]. Available: http://developers.google.com/chart/

[38] E. H. Land and J. McCann, "Lightness and retinex theory," *JOSA*, vol. 61, no. 1, pp. 1–11, 1971.

[39] D. J. Jobson, Z.-U. Rahman, and G. A. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *Image Processing, IEEE Transactions on*, vol. 6, no. 7, pp. 965–976, 1997.