Find Birds' Nests with Image Processing

Shengdong Liu and Xiangyun Zhao

Abstract

This paper describes a computer vision approach to help ecologists find bird nests. The work involves processing two types of aerial images of an area taken by quad-copters. Near infrared images are used to detect and exclude plants from the search area. Visible light images are classified to find trees and to exclude irrelevant landscapes such as soil, rocks, and water. When used alone, near infrared images can eliminate roughly 50% of the search area, and visible light images can eliminate up to 89% of the search area. This is a promising first step to help ecologists find bird nests by combining many layers of information from different types of images to effectively reduce the search area.

1 Introduction

Birds are important to the ecosystem. For the animal kingdom, birds are in many layers of the food web, so they help to maintain the population of their preys and predators. For the plant kingdom, birds are the pollinators and seed dispersers, so they help with the reproduction of plants. Their presence in an ecosystem has undeniable impact for many other species, directly and indirectly[1]. In addition, there are over 10,000 types of identified birds[2] and their presence are widespread around the world. These traits make birds common yet fascinating, and attract many ecologists to specialize in studying birds.

Though fascinating, the process of studying the birds is not all excitement, especially when the ecologists cannot find the target birds to study. Even if the ecologists found right type of birds, it is difficult for ecologists to observe the birds for a significant period of time because birds are highly mobile. Of course, with the advancement in technology, we could, in theory, track the birds with aerial vehicles, but that would be highly impractical in term of cost. Thus, most ecologists who specialize in birds would set up cameras in the areas where birds commonly appear, and hope the cameras would capture the birds' activities. This approach is practical, but the success of this method relies heavily on the cameras' locations. In other words, the cameras needs to be set up at locations where there are birds flying by for the method to be effective. This leads to another major inconvenience in bird studies, that is how to locate areas where birds are commonly seen. Ecologists can, with out much difficulty, find the appropriate locations for a large population of birds that fly in masses. However, what about the endanger birds that are scarce but need more attention and protection? The most practical way is to find the birds' nests, and set up cameras to monitor where the nests are. That way, ecologists will be able to see the birds every time they leave or return to their nests. However, but the process of finding the birds' nests is very tedious and inefficient. Currently, ecologists have to go around in person to find the birds' nests with out much guidance. This is problems makes the preparation for studying birds very time consuming, and severely limits the productivity of ecologists who would like to observe and help the birds.

Engineers For Exploration (E4E) is a group of students who develop and use technology to drive the future of exploration[3]. With the the goal to apply remote imaging, sensing, and robotic technologies to extend the limits of human exploration, the group accepted the challenge ecologists have to face in finding birds' nests. The goal for this project is to help ecologists find birds' nests efficient. To take advantage of the technology, a quad-copter will be used to scan a landscape because it has high mobility. To track birds with quad-copters would be impractical. Quad-copter is limited by weight, and thus cannot carry a large supply of power to keep it in the air for a long period of time. However, the flight time to survey landscape should be relatively short and the battery should suffice. On the quad-copter, there are a few different camera systems that take photos of the landscape. These systems include visible light camera, near infrared camera, and thermal camera. The purpose of putting all these systems on the quad-copter is to get many layers of information from the landscape, and then, these images can be analyzed to produce informative data to help ecologists look for birds' nests.

This paper will cover mainly three steps taken toward helping ecologists with finding the birds' nests, and the methodology and result of each step taken:

- The results of processing the near infrared images by finding the Normalized Difference Vegetation Index (NDVI).
- The results of processing the visible light images by using machine learning algorithms to classify the different parts of the images to find where the trees are.
- The analysis of the acoustic impact that quad-copter to stay under to not disturb the wildlife.

2 Methodologies

2.1 Near IR Images

The raw images we have access to are NGB images, images that has it's red value of the RGB channels replaced by infrared. We extracted the near infrared value from the red channel and calculated the NDVI (Normalized Difference Vegetation Index) using the following formula:

$$\mathrm{NDVI} = \frac{(\mathrm{NIR} - \mathrm{VIS})}{(\mathrm{NIR} + \mathrm{VIS})}$$

The VIS and NIR stand for the measurements acquired in the visible and near-infrared regions. Typically, the red value is used for the visible, but since our NGB images did not have a red channel, we used the blue for the visible region, which should yield similar result. Because the index is calculated using ratios of blue light absorbed versus IR light reflected, photosynthetic materials will stand out more. To put it simply, NDVI measures the health of plants [4]. Since birds' nests are made of dead twigs, and the surround environment is usually leaves that keeps the nest hidden, NDVI can be use to distinguish the nests and the leaves.

We tested this method on NGB images of wetland from taken from San Elijo and here's a example of what we have and the NDVI image generated.

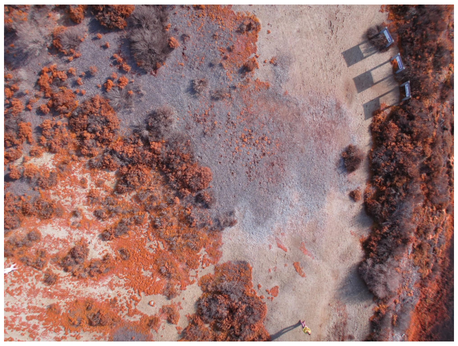


Illustration 1: NGB Image

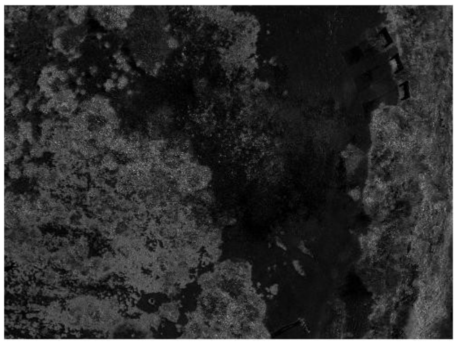


Illustration 2: NDVI Image

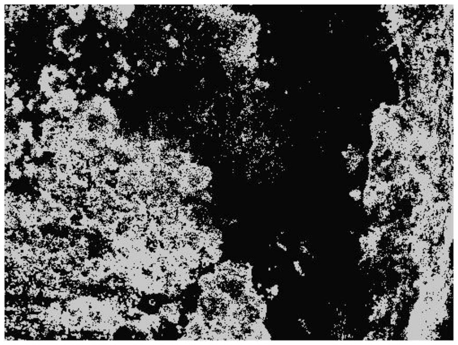


Illustration 3: Binary Image

Illustration 1 shows the raw NGB image we got, and Illustration 2 is the NDVI Image we generated by calculation. Illustration 3 is a binary image generated from the NDVI image by setting and threshold to distinguish the life and dead material. As we can see, the plants appears as white pixels on the binary image, and the non-photosynthetic material are dark. This is the first layer of information we can use to reduce the possible search areas by eliminating the plants. Form the images we tested on, we can eliminate about 50 percent of the landscape on average, since about half the area are photosynthetic materials such as leaves and grass, and about half of the landscape are rocks, soil and water. This leads us to work on next part of the study, which is how to eliminate these other dead materials that's not a birds nest?

2.2 Visible Light Images

2.2.1 Neural Network Classification

2.2.1.1 Collect Data and Data Preprocessing

We fly the copter in the filed to collect many field images. The images are like the image following.

Then we manually collect 250 by 250 data images for each class(leaves, rock, soil, grass). The data images like following:

Grass Leaves Rock Soil

Because we do not have access to GPU, so we resize the 250 by 250 images to 28 by 28 images. Before that we transform the RGB image to gray scale image. The smaller data images will reduce the train time significantly. And we convert each image to one vector with 784 elements. After doing this for whole data(we manually collected 2214 images), we calculate the mean vector, and the mean standard deviation. Then we can convert the original vectors to 0 mean and 1 variance. This part is in MATLAB function pre_data.

2.2.1.2 Neural Network Architecture and Approach

2.2.1.2.1 Cost and Gradient Function Design

We wrote a function cnnCost which computes the cost, gradient and predicted classification using a given set of weights on a given data set for a specified architecture. We define the architecture in the form of a cell structure, whose rows correspond to layers. For a given layer, the row contains a string with the layer type: either ”pooling”, ”convolutional”, or ”connected”. For a convolutional layer, the row also contains a number specifying the number of filters, and for a pooling layer, the row also contains a number specifying the pooling dimension. Finally. for all three types, the row contains another string specifying the options for the layer. In a convolutional or fully connected layer, this would be the activation function, and in a pooling layer, this would be either mean or max pooling. The activation functions we implemented are "funny tanh", "leaky relu", "sigmoid", and "softmax".

We also pass in the weights as a cell structure. The rows correspond to layers. Each row has a weight and bias component. For pooling layers, the weight and bias are empty matrices. For convolutional layers, the weight is a four dimension matrix whose dimensions are the filter size (twice), the dimension of filters which should equals the number of filters in the previous convolutional layer (or one if there wasn't one) and the number of filters. The bias is a vector whose length is the number of nodes in the next layer.

Our function computes the cost and gradient with several helper functions. For each of the three layer types, we there is a forward propagation and a back propagation function. The forward propagation takes the previous layer of nodes as input and outputs the next layer of nodes. The back propagation function takes the deltas from the previous step, which have already been multiplied or convolved with the weights, and outputs the next deltas as well as the gradient for the weights in that layer.

Our function computes the cost by iterating through the layers and calling the appropriate forward propagation function for each layer, and then computing the cost after the last layer. Then it iterates backwards through the layers and calls the appropriate back propagation function, building the gradient as it goes.

The forward and backward propagation functions for pooling, convolutional, and fully connected layers are described below.

2.2.1.2.2 Pooling

To forward propagate in mean pooling with pool size D, for each image, we iterate through all copies of the image that have been generated by convolutional layers. For each copy, We divide this image into small patches whose dimension equals the pool size, we do a valid convolution of this patch with a square matrix whose dimension is D and whose elements are $\frac{1}{D^2}$.

To forward propagate in max pooling, we iterate through all copies of the image that have been generated by convolutional layers. For each copy, we simply find the maximum element in each non-overlapping square of dimension D by D, and copy this element into a new matrix. And remember this maximum element location in each copy.

To back propagate for mean pooling, we iterate through all copies of each image. For each copy, we take the Kroneckor product of the delta matrix with a square matrix of ones of dimension D, then divide the whole thing by D^2. This has the effect of upsampling the errors.

To back propagate for max pooling, we do the same thing except that this time we do not divide by D 2. Then after upsampling, we need make sure that all elements of the delta matrix are zeros except for the ones that were the max in their region.

The output of back propagation is the new matrix of deltas, along with the gradient which is empty in this case since there are no weights in a pooling layer.

2.2.1.2.3 Convolutional Layer

For forward propagation in the convolutional layer, let us suppose we have m copies of each image from the previous convolutional layer, and n new filters. For each image, we iterate through the copies and through the filters. For the i-th copy and the j-th filter, we take the filter at index i, j in the weight matrix, rotate it, and do a valid convolution of this with the corresponding copy of the image.

Then for each filter, we sum over the m copies and add the bias term corresponding to that filter.

$$M_j^{new} = \sum_i M_i * W_{ij} + bias$$

Thus we now have n convolved copies of each image. Now, we apply the activation function to each element a of the new matrix. For sigmoid, this is:

$$g(a) = \frac{1}{1+e^{-a}}$$

For funny tanh, this is:

$$g(a) = 1.7169 \tanh(\frac{2}{3}a)$$

For leaky relu, this is:

$g(a) = a(a > 0)$ otherwise, g(a) = 0.01a;

For back propagation, we first compute the gradient of the activation function.

For sigmoid, this is: $g(a)' = g(a) \ - g(a)^2$

For funny tanh, this is: $g(a)' = 1.7169 \times \frac{2}{3}(1 - \frac{1}{1.7169} g(a))$

For leaky relu, this is: $g'(a) = 1(a > 0)$, other wise $g(a)' = 0.01$

We then compute the deltas using the deltas passed in from the previous step. These deltas have already been multiplied by the weights, so we simply compute:

$$\delta_{new} = g' \times \delta_{weighted}$$

We compute the gradient using these new deltas. To compute the bias gradient we sum over all input images and sum over the rows and columns of each image. To compute the weight gradient, we iterate through the n filters and the m previous copies. For the i-th copy and the j-th filter, we do a valid convolution of delta matrix corresponding to that filter with the image from the layer below corresponding to the i-th copy:

$$\delta_j^{weighted} = \sum_i \delta_i * W_{ij}$$

The output of this back propagation step is the weighted deltas and the gradient[5].

2.2.1.2.4 Fully Connected

To forward propagate in a fully connected layer, we multiply the matrix of previous nodes times the weight matrix and then add the bias vector to each row:

Y =XW+B

where X is the matrix of the previous nodes, Y the matrix of the next nodes, W the weight matrix, and B a matrix whose rows are copies of the bias vector. Then we apply the activation function as we did in the convolutional layer. We also have the option to do softmax activation here, which is:

$$g(Y_j) = \frac{e^{Yj}}{\sum_i e^{Y_{ij}}}$$

Where Yj is the is the j th column of Y

To do backward propagation, for activations other than softmax, we compute g' as we did in the convolutional layers. Then we compute the new delta using the previous delta which was passed in (which has already been multiplied by the previous weights.)

$$\delta_{new} = g^{'} * \delta_{weighted}$$

Softmax activation only occurs on the last layer. We assume cross entropy loss. In this case, we pass in the labels instead of the previous deltas. We make a matrix T whose rows correspond to input images, and whose columns correspond to classes, containing a 1 in a row and column if that input is in that class and a zero otherwise. Then the matrix of deltas is: $\delta = O - T$ where O is the final output of the softmax layer.

The gradient for the weights is computed: $g = X\delta_{new}$ where X is the matrix of nodes before these weights were applied. The gradient for the bias is the sum of the deltas over the input images.

Finally, we multiply the deltas times the weights to pass to the next layer: $\delta_{weighted} = \delta_{new} * W$ [6].

2.2.1.2.5 Architecture

The architecture is:

input vectors -> convolution 5 9*9*1 filters -> convolution 10 5*5*5 filters -> max pooling layer -> fully_connected layer 30 hidden unit -> softmax layer 4 output.

2.2.1.2.6 Stochastic Gradient Descent

For this part, we did the minibatch gradient descent method. When we minimize the cross entropy function, a standard (or ”batch”) gradient descent method would perform the following iterations :

$$weight_{updated} = weight - \alpha \times gradient$$

where α is a step size (sometimes called the learning rate in machine learning).

In many cases, evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions’ gradients. The minibatch stochastic gradient descent is to randomly choose one minibatch of the whole data to do the gradient descent instead of using the whole batch, it will be more fast and economic because the whole data is too tremendous. Here we choose minibatch number to be 256. After choosing the minibatch, we calculate the gradient and cost just based on the random minibatch each time. Then update the weight and bias. Another trick we used h ere is to use the momentum to increase the convergence time. The learning rule with momentum is:

$$V_{t+1} = \mu V_t - gradient$$

$$W_{t+1} = W_t + \alpha V_{t+1}$$

We we chose $\mu = 0.02$ and $\alpha = 0.01$. Momentum makes the path towards the minimum smoother. The number of iterations is 3000. Our only stopping criterion is the number of iterations.

2.2.1.2.7 Test

Although the input images are much smaller than the original data images, we still spent 5 hours on training on CPU. The architecture is in the MATLAB file cnnTrain.

After training the neural network, we started to do the test. We arbitrarily choose the test images from the field images. Then divide the whole image to 250 by 250 small patches and do the classification. After cutting out the each patch, we resize it to be 28 by 28. Before resizing the image, we transform the RGB image to gray scale image. Then convert the the image to be one vector with 785 elements. Then subtract the mean value (which is obtained from the training data) and divide the standard deviation (which is obtained from training data). For the classification part, the code is in the MATLAB test file. Because what we need to do is just exclude the areas which are not the leaves and the resolution is not the problem, it is unnecessary to do the pixel classification. What we need to do is to do the patch classification. And then The results will be the following parts.

2.2.2 Local Binary Pattern Feature Classification

Given a pixel in the image, an LBP code is computed by comparing it with its neighbors:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,\ s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

where gc is the gray value of the central pixel, gp is the value of its neighbors, P is the total number of involved neighbors and R is the radius of the neighborhood. After the LBP pattern of each pixel is identified, a histogram is built to represent the texture image:

$$H(k) = \sum_{i=1}^{I} \sum_{j=1}^{J} f(LBP_{P,R}(i,j),k), k \in [0,K],$$

$$f(x,y) = \begin{cases} 1, x = y \\ 0, otherwise \end{cases}$$

[7]

we define the above operator as the CLBP_S operator. Then we define another operator as following: CLBP_M operator

$$CLBP\_M_{P,R} = \sum_{p=0}^{P-1} t(m_p, c)2^p,\ t(x,c) = \begin{cases} 1, x \geq c \\ 0, x < c \end{cases}$$

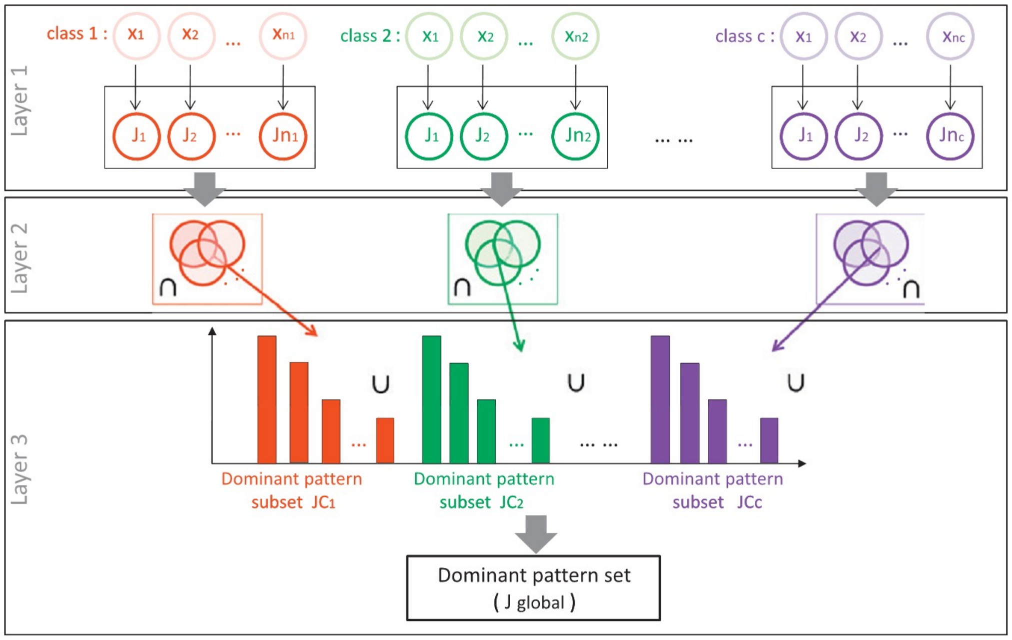
where c is a threshold to be determined adaptively. Here we set it as the mean value of mp from the whole image. We calculate the histograms of the CLBP_S and CLBP_M codes separately, and then concatenate the two histograms together[8].

Dominant pattern set of an image is the minimum set of pattern types which can cover n (0 o n o 1) of all patterns of an image. This can be expressed by the following equation in order to find a set Ji for image

$$J_i = \arg\min_{|J_i|}\left(\frac{\sum_{j\in J_i} f_{i,j}}{\sum_{k=1}^{p} f_{i,k}}\right) \geq n,$$

where $|J_i|$ denotes the number of elements in set Ji ($J_i \subseteq [1,2...., p]$), $f_{i,j}$ is one of the patterns for class i.

To construct a discriminative and robust pattern set, the learning model contains three layers for each target: feature robustness, discriminative power, and the capability of representation. The framework is as following：



For each layer the algorithm is described as following:

Algorithm 1. (*Layer* 1) Get the dominant pattern set of each training image x_i from the training set $\mathbf{T}_{train}$.

Input: The histogram $\vec{f_i}$ of the original pattern sets of interest of each training image x_i, and the threshold parameter n to determine the proportions of dominant patterns selected from each training image.

Output: Dominant pattern set J_i with respect to each training image x_i.

```
1. Initialize reference vectors $\vec{V}$, where $\vec{V}[i]=(i-1)$
   ($i=1,...,p$), where $p$ denotes the number of all possible
   patterns types of interest.
2. Sort $\vec{f}_i$ in descending order, resulting in $\hat{\vec{f_i}}$. Change the
   configuration of $\vec{V}$ according to element order of $\hat{\vec{f_i}}$. The
   resulting vectors are denoted as $\hat{\vec{V}}$.
3. FOR $k=1$ to $p$
```

```
4.    IF $\left( \sum_{l=1}^{k} \frac{\hat{\vec{f}}_{i,l}}{\sum_{l=1}^{p} \hat{\vec{f}}_{i,l}} \geq n \right)$
5.        BREAK
6.    END IF
7. END FOR
8. Return $J_i = \{\hat{\vec{V}}[1], \ldots, \hat{\vec{V}}[k]\}$ as a dominant pattern set for
   image $x_i \in \mathbf{T}_{train}$.
```

Algorithm 2. (*Layer* 2) Estimate the discriminative dominant pattern set of each class j.

Input: Dominant pattern sets $J_1, J_2, \ldots, J_{n_j}$ of n_j images belonging to class j obtained from Algorithm 1.

```
Output: Discriminative dominant pattern set $JC_j$ of class $j$.
1.    Initialize $JC_j = J_1$.
2.    FOR each image $k=2$ to $n_j$ belonging to class $j$
3.        $JC_j = JC_j \cap J_k$.
4.    END FOR
5.    Return $JC_j$.
```

Algorithm 3. (*Layer* 3) Construct the global dominant pattern set.

```
Input: The discriminative dominant pattern set $JC_j$ for each
class $j$ $(j = 1, \ldots, C)$ obtained from Algorithm 2.
Output: The global dominant pattern set $J_{global}$.
1.    Initialize $J_{global} = \emptyset$.
2.    FOR $k=1$ to $C$
3.        $J_{global} = J_{global} \cup JC_k$.
4.    END FOR
5.    Return $J_{global}$.
```

[8]

After extracting the dominant patterns, we extract the corresponding features from the training images. And then use the nearest neighbor method to do the classification. It means that find the minimum L_1 distance between the test feature and the train feature. The corresponding train feature label is the test label.

2.2.3 Results and Discussion

The leaves area is the potential bird nest area, we call it positive area, other areas are all negative areas. Then the results are following:

Neural Network method:

- True Positive 90.4% false positive 9.6%
- True negative 99.99% False negative 0.01%
- Exclude the 89.94% area which can not be bird nest

From the results, although there is still some false positive, the false negative percent is very low. It means that the almost all the potential bird nest area can be detected. Because we resize the original data to much smaller images.. The performance is definitely influenced. Further more, because the data is limited, we can not implement more complex architecture. If we implement more complex architecture, there will be some over-fitting. So I think, if we have access to GPU to deal with the original 250 by 250 data image, and have more data, the performance will definitely be improved.

LBP feature method:

- True Positive 90.7% false positive 9.3%
- True negative 99.98% False negative 0.02%

From the results, the traditional method and the neural network method seem to have the similar performance, but actually, if we have more data and can access to GPU, the neural network method will definitely perform better.

2.3 Acoustic Impact

Although quad-copter is highly mobile and ideal for surveying the landscape, it is not without drawbacks. Quad-copter creates noise, and the noise emission during flight could potentially disturb the birds and other animals in the area. Thus we need to keep the quad-copter at a safe altitude so that the noise will not be a disturbance to the wildlife while operating at a elevation where high quality images can be taken. This section will discuss the guidelines we have to operate the quad-copter with out disturbing the animals in the area.

According to Federal Aviation Association (FFA) Regulation and Guidance on visual flight rules near noise-sensitive areas, aircraft pilots are encouraged to voluntarily practice flying of 2,000 feet above noise sensitive areas such as National Parks, Nation Wildlife Refuges, water fowl Production Areas and wilderness Areas[9]. Of course, this is not applicable to quad-copters as the FAA safety also limit recreational use of model aircraft to below 400 feet[10]. So, there's no direct elevation guideline for quad-copters.

A more roundabout approach we have to resort to would be to measure the sound of a quad-copter in a unit such as decibels, and then comparing that with data that we already have regulations on. The FAA says that a maximum day-night average sound level of 65 dB is incompatible with residential communities[11]. This is a regulation in regards to human, but fortunately for us, for most birds has roughly similar hearing range to human hearing[12]. Thus if we can show, statistically, that our quad-copter operates on average less than 65 dB, it should be relatively safe for the animals in the area. Thus to make the quad-copter deployable, our immediate future goal is to collect acoustic data on the noise level of our quad-copter in operation, and find the operating level that will meet the requirement.

3 Milestones

To summarize our milestones and progresses, we started with 3 major milestones: process near infrared images, process thermal images, and write up acoustic impact report. We completed 2 out 3 major milestones: process near infrared images, and write up acoustic impact report. We decided to drop the milestone for processing thermal images and reallocate that time to work on a machine learning approach to classify the visible light images. The reason we decided to drop thermal images processing in favor of machine learning is that we intended to implement classification also on near infrared images. That way we can overlay the two layers of information using only one type of image with out knowing needing to know the exact location where the image is taken. Classifying the near infrared image however, is disappointingly inaccurate because the lower resolution lead to lost in texture. This caused us to revert back to classifying on the visible light images. When we have a way to match the location of all the different types of images, we can overlay the information to produce better results. The following table shows in detail each of the milestones we had. (Note: for all the milestones without the key word "New", those are the original milestones we had at the beginning of the quarter, the milestones with the key word "New" are milestones that we inserted later on in the quarter.)

Milestones	Description of Deliverables	Due Date	Completion Time/Status	Comments
Get images from NGB camera	NGB images (e4e.ucsd.edu/eric/san_elijo.tar.gz)	End of WK 4	WK 4	Provided by Eric
Process NIR images using NDVI	NDVI images (See WK4 Update https://github.com/shl202/BirdNest/wiki/Weekly-Updates)	End of WK 4	End of WK 4	Not satisfactory, binary image from threshold the NDVI image returns ~50% white and ~50% black
New: Approach to further reduce search area.	Methodology Proposal (See WK5 Update https://github.com/shl202/BirdNest/wiki/Weekly-Updates)	End of WK 5	End of WK 5	See Week 5 update on our wiki page.
New: Collect Training Data	Classified training data for water, rock, grass etc.	End of WK 6	End of WK 7 (Late)	We have 537 images total as of now, and it is working for the classification method, although more samples the better.
New: Classification by texture and train.	Classification Report https://github.com/shl202/BirdNest/blob/master/doc/Classification%20Report.pdf	End of WK 6	End of WK 7 (Late)	The accuracy is good based on our current samples
Get and analyze thermal images	Thermal images	End of WK 7	Dropped	Decided to reallocate the time on classification instead, which is more promising.
Integrate thermal analysis into our search area	Report of how thermal images helps	End of WK 8	Dropped	Decided to reallocate the time on classification instead, which is more promising.
New: Classify NIR images	Classify using NIR images for easier combining process with NIR result	End of WK 8	Aborted	Testing the NIR images with our classification method, gave us inaccurate results due to lower resolution of NIR images.

Acoustic impact analysis	Acoustic impact analysis, see Section 2.3	End of WK 8	WK 9 (Late)	Found some guidelines for the acoustic level we should aim measurements from Eric for the acoustic impact of our quad-copters.
New: Neural-Network Approach for Classification	See Demo Video from 5:34 – 8:25 (https://www.youtube.com/watch?feature=player_detailpage&v=JfbsgYltWcQ#t=334)	End of WK 10	End of WK 10	This method is more promising with accurate as we can improve it more with more training data
New: Collect Train Data	Training images https://github.com/shl202/BirdNest/tree/master/classified%20samples/250x250	End of WK 10	WK 10	Collected training images for 1000 leaves, 500 rocks, 500 grass, 200 soil.
Project Video	Video Demo https://www.youtube.com/watch?v=JfbsgYltWcQ	End of WK 10	WK 11 (late)	See Video Demo!

4 Future Work

The results from this work contributes to the initial steps of finding birds' nests. For the future, there are many more steps to come before helping ecologists to efficiently find the birds' nests. Thermal images can be processed to find the animals in the landscape. The difference in the animal's body temperature and their surrounding can be quite noticeable, so processing the thermal images can give us another layer of information that further narrows down the possible area where the birds' nests might be. An important future work is to bring all the pieces of information together. To do that, we need the geographic coordinates of the images so that for any near infrared image we process, we can find the corresponding visible light and thermal image and vice versa. Finally, generating a shortest path to guide the ecologists to all the potential areas would increase the usability of this project for the ecologists to find the birds nest efficiently.

5 Conclusion

This work provide the initial steps to find birds' nests using image processing. Two types of images, near infrared and visible light, were processed and the results show they can climate up to 50% and 89% of the search search area respectively. This work also provided some insight on the noise level (65 dB) the quad-copter should operate under so the animals are not disturb by the acoustic impact. These foundational steps will be useful for helping ecologists find birds' nests more efficiently.

Reference

[1] "Ecological Roles of Birds" Endangered Species International, 2011. Retrieved 10 June 2015. <http://www.endangeredspeciesinternational.org/birds4.html>

[2] Gill, Frank (2006). *Birds of the World: Recommended English Names*. Princeton: Princeton University Press.

[3] *Engineers For Exploration*, 2015. Retrieved 10 June 2015. <http://e4e.ucsd.edu/wordpress/?page_id=2>

[4] Liz. "NDVI and NRG" Public Lab, 2013. Retrieved 10 June 2015. <http://publiclab.org/wiki/ndvi>

[5] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton "ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems 25 Year: 2012 Pages:

[6] Online Stanford Deep Learning Tutorial UFLDL Tutorial Website: http://ufldl.stanford.edu/tutorial/

[7] Yimo Guo , Guoying Zhao , Matti Pietikainen, "Discriminative features for texture description" Pattern Recognition 45 (2012) 3834–3843

[8] Zhenhua Guo, Lei Zhang and David Zhang, A Completed Modeling of Local Binary Pattern Operator for Texture Classification",IEEE Transactions on Image Processing

[9] "Visual Flight Rules (VFR) Flight Near Noise-sensitive Areas" Federal Aviation Administration, 17 September 2004. Retrieved 10 June 2015. <http://rgl.faa.gov/Regulatory_and_Guidance_Library/rgAdvisoryCircular.nsf/list/AC%2091-36D/$FILE/AC91-36d.pdf>

[10] "What Can I Do With My Model Aircraft?" Federal Aviation Administration, 12 August 2014. Retrieved 11 June 2015. <http://www.faa.gov/uas/publications/model_aircraft_operators/>

[11] "Noise Monitoring". Massport. Retrieved 10 June 2015. <http://www.massport.com/environment/environmental-reporting/noise-abatement/noise-monitoring/>

[12] Beason, C., Robert. "What Can Birds Hear?". *USDA National Wildlife Research Center - Staff Publications*. Retrieved 10 June 2015. <http://digitalcommons.unl.edu/cgi/viewcontent.cgiarticle=1076&context=icwdm_usdanwrc>