Distribution Agreement

Zhixin Tan April 15, 2018

Video tracking and identifying unmarked moving insects

by

Zhixin Tan

Dr. Ilya Nemenman
Adviser

Department of Biology

Dr. Ilya Nemenman

Adviser

Dr. Gordon Berman

Committee Member

Dr. Berry Brosi

Committee Member

2018

Video tracking and identifying unmarked moving insects

by

Zhixin Tan

Dr. Ilya Nemenman

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2018

Abstract

Video tracking and identifying unmarked moving insects
by Zhixin Tan

The quantitative measurement of animal behavior is a critical problem in many biological studies. Computer-assisted video tracking is among the most prominent methods for the measurement, especially for small animals such as insects. However, many existing video tracking methods cannot identify unmarked insects efficiently, which compromises the tracking quality. In this study, we developed a framework that involves Kalman filter, color correlogram comparison (idTracker), and artificial neural networks to track and identify multiple insects in the video, even when their trajectories are interrupted due to occlusion. We implemented and tested these algorithms on the videos of bumblebees, several of which showed desirable performances. We also evaluated the performance of different individual insect identifiers and proposed some directions for improvement. This study demonstrated the feasibility of this framework and supported the possibility of being widely used in insect behavior research.

Video tracking and identifying unmarked moving insects

by

Zhixin Tan

Dr. Ilya Nemenman

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2018

Table of Contents

Video tracking and identifying unmarked moving insects

Cyrillus Zhixin Tan

26 March 2018

The quantitative measurement of animal behavior is a critical problem in many biological studies. Computer-assisted video tracking is among the most prominent methods for the measurement, especially for small animals such as insects. However, many existing video tracking methods cannot identify unmarked insects efficiently, which compromises the tracking quality. In this study, we developed a framework that involves Kalman filter, color correlogram comparison (idTracker), and artificial neural networks to track and identify multiple insects in the video, even when their trajectories are interrupted due to occlusion. We implemented and tested these algorithms on the videos of bumblebees, several of which showed desirable performances. We also evaluated the performance of different individual insect identifiers and proposed some directions for improvement. This study demonstrated the feasibility of this framework and supported the possibility of being widely used in insect behavior research.

1 Introduction

The behavior of animals is one of the most important phenotypes to be observed and measured in many biological studies, such as genetics and neurobiology. However, a prominent technical issue such research has been encountering is to quantitatively measure and model the behaviors of animals. Many experiments on insects, such as flies, bees, and mosquitos, are operated in a confined and controlled space in the laboratory, and quantitatively measured data on the movements and behaviors of these insects can provide valuable insights for these experiments.

Due to its cost efficiency, accuracy and versatility, video tracking is one of the primary solutions for tracking animal movements. Computer-assisted video tracking has multiple benefits over many other methods, including reliable recording, computer algorithms ensuring stable performance, and consistent accuracy [1].

However, video tracking on small insects faces various challenges. First of all, video data themselves are not directly quantitative and descriptive information before they are processed and analyzed. For small flying insects, the analysis can be more difficult because freely moving insects can change their appearances in the video over time [2]. When the insects move slowly and sparsely, the video analysis can be relatively easier. Some classical tracking methods, including the Kalman filter, provide a solid foundation for such cases and have been widely applied [3]. However, when the insects are close, and when they are touching, occluding, or crossing over each other, it is more difficult to keep track of all of their trajectories and identities [4].

One possible solution to this problem is to develop a method to distinguish one individual from another. In this way, even after an interruption of a trajectory, we can still identify the individual and continue the tracking once it is visible again.

Many methods and frameworks have been proposed and developed to distinguish individual insects of the same species from each other [5]. However, many methods require the insects to be tagged [6], and tagging the insects can be intrusive and can influence their behaviors. Among the existing methods of identifying unmarked animals, the idTracker, developed by Perez-Escudero *et al.* [7], is one of the more recent successful approaches. This method extracts the color correlogram of the image of an animal as the features to characterize the individual. It has been demonstrated to work on fish, flies, ants, and mice, but there is no published work about implementing it on bumblebees tracking.

Another approach to identify the individuals is based on machine learning. The idea of artificial neural networks and "deep learning" is one of the most successful and fastest developed approaches to machine learning. Inspired by biological neural systems, this method has been widely applied to many areas of artificial intelligence [8]. An artificial neural network consists of multiple layers of perceptrons: an input layer, an output layer, and several hidden layers in the middle. Each layer contains multiple units that recombine and calculate the inputs from the last layer. The parameters of these units are learned from the input data following a general-purpose learning procedure [9]. The convolutional neural network is a specialized kind of neural network for processing image-like data by learning convolutional kernels iteratively [10]. This technique has performed outstandingly in image classification and identification due to its strong ability to recognize complex features from image data or any grid-like data. However, this method has rarely been implemented to identify flying individual insects of the same species.

In this study, we used bumblebees, *Bombus impatiens*, as a model organism, tracking them in the experimental setting and keeping the identity of each individual. We implemented, tested, and evaluated the video tracking and identification methods, including the Kalman filter, idTracker, and artificial neural networks. We develop a framework for tracking multiple unmarked insects in a confined area. Based on the video tracking technique, this framework first preprocesses the videos capturing the movement of the insects and detects each individual insects in the frame. When the insects fly rarely and their trajectories do not interfere with each other, we apply classical tracking methods such as the Kalman filter to track them over time. For each individual insect participating the experiment, we construct a reference set, or training set, from the video recorded during the experiment. These training sets are used for classification programs, such as the idTracker or the artificial neural networks, to learn the features of each individual in order to classify an unidentified insect. In the case where a trajectory of an insect crossed another, an insect disappears from the frame, or a new insect appears into the frame, using the predeveloped classifier, the framework can identify which individual it is and restart its tracking. We believe this framework will generate reliable tracking information and provide the insect researchers a powerful tool to study insect behaviors.

2 Material and Methods

Experimental setup

All experiments performed in this study were inside a spacious flight chamber with proper lighting from the top, as shown in Figure 1. The bottom of the chamber was made of white acrylic sheets. Since the bees were of dark colors, the white floor helped to increase the contrast between the bees and the background. Additional experimental equipment used for other experimental purposes was fixed on the white floor, such as artificial flowers in blue or yellow color and LED lights, which were not of importance to this study. A transparent plastic sheet of 24-inch x 24-inch was used to confine the bees to a relatively flat space close to the floor while allow the movement of bees to be observed from above. The transparent plastic sheet was raised to 1 inch above the white acrylic sheet floor by several plastic columns. Metal meshes were attached around the four edges of the transparent plastic sheet to enclose the bees while allowing adequate air flow. During the experiment, a bee or multiple bees of interest were released on the white acrylic sheet and covered by the transparent plastic sheet.

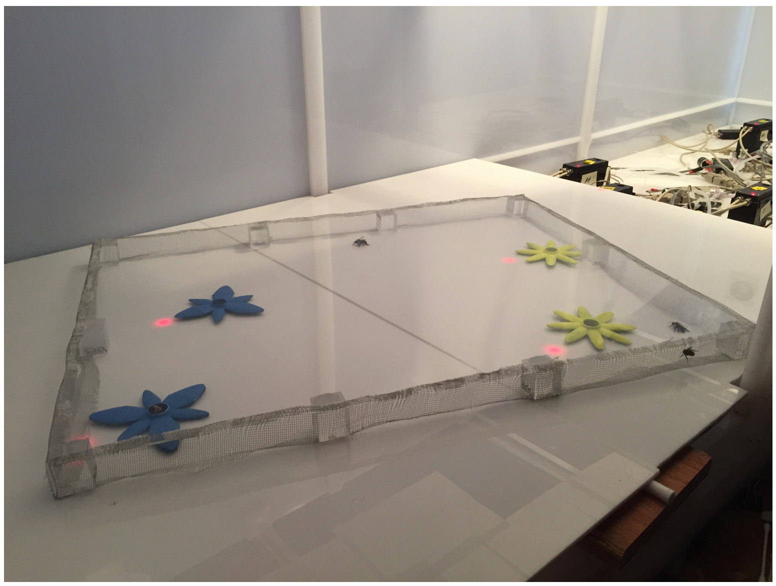


Figure 1: The experimental setup

A webcam was placed above the described setup, pointing to the bees moving area, to record videos of their motions. It was a Logitech HD Pro Webcam C920 which captures videos of 1280×720 pixels resolution in 25 frames per second. During the video recording, the position of the webcam was fixed to ensure consistent reference frame among various videos. The recordings were saved as files for further processing. Unless specified, all computer programs used to analyze the videos in this study were written in Python 3.0, with the Open Source Computer Vision Library, OpenCV.

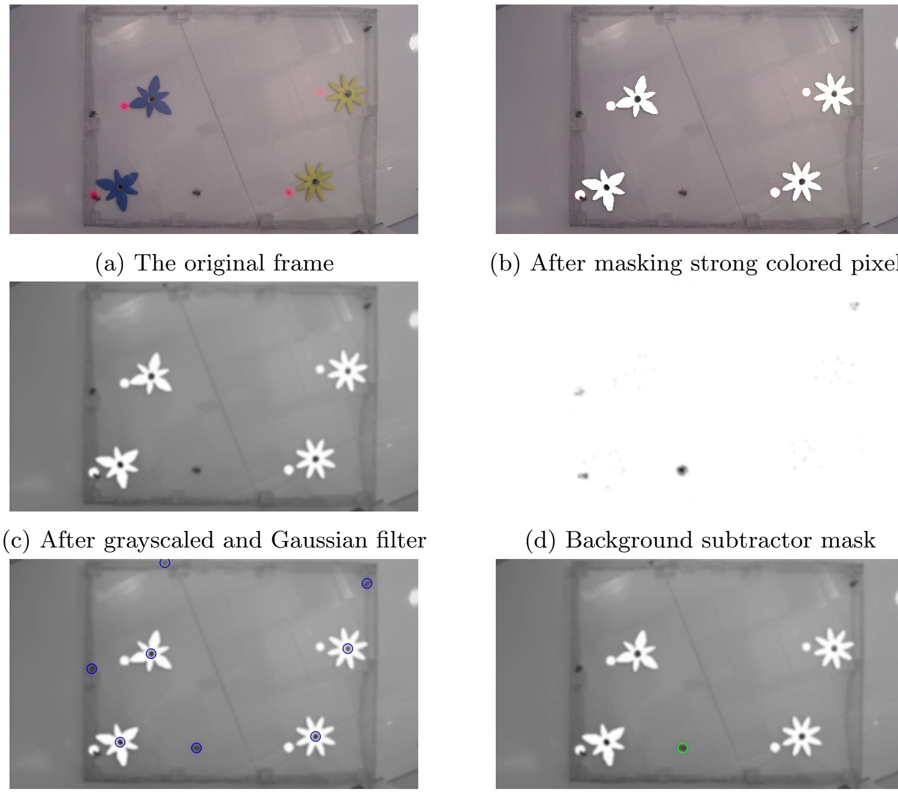
Video Preprocessing

The collected videos were first analyzed by a preprocessing program that found the bees in the frames, as shown in Figure 2. To reduce the interference of the additional experimental equipment, such as artificial flowers and LED lights, pixels with strong color intensities were first masked. Strong colored pixels were identified as those with total intensity difference among three color channels (blue, green and red) above a certain threshold (in our case, 40 out of the range of channel 256) which depended on the lighting condition during the video shooting process.

To recognize bees in the video, the program looked for small (3 to 6 cm long), dark blobs sitting on a white background. The frame was converted into grayscale to simplify the image processing procedure. Since the shapes of bees appeared irregularly concave in the video (consider the outstretching legs, antenna, and wings), to enhance the performance of the blob detector, a Gaussian filter was applied to each frame to smooth out the perimeter.

Another approach to recognize whether an object is a bee or not is to look for the moving elements in the frame. For this purpose, a background subtractor was applied. However, since any small lighting change will result in changes of background from one frame to the next, artifact detections would occur and, hence, it was not sufficient to detect bees. However, by combining the blob detection and motion detection with an AND gate, the program will only report the blobs situating on the areas where the background changes, and a more reliable bee detection could be achieved.

Finally, we need to construct the training sets for the profiles of individual bees for the application of the machine learning methods. For this purpose, we released each bee separately to the flight chamber and recorded an individual bee video. The preprocessing program could be used to recognize the bee and cropped the region of detection to make a collection of images for this bee. The cropped images were of the size of 40×40 pixels among which the actual bee occupied approximately 250 pixels. The data were all inspected and cleaned up manually to eliminate fuzzy and wrong data points from the training sets. Some sample images of the bees are shown in Figure 3.



(a) The original frame

(b) After masking strong colored pixels

(c) After grayscaled and Gaussian filter

(d) Background subtractor mask

(e) Blob detector results

(f) Blob and motion detector results

Figure 2: Frames after each step of preprocessing

Trajectory tracing and Kalman filter

For tracking the trajectories of the bees from the video data, the center point of each detected blob is defined to be the measured position of that corresponding bee. An issue of tracking a bee over time is to determine which detected bee in the last frame is the identical to the ones in the current frame. To map the bees in two consecutive frames, we assume that the two blobs with closest distance in two adjacent frames belong to the trajectory of the same bee. However, this simple approach is challenged by situations where a blob from the last frame is no longer detected, or a new blob appears in the current frame, or the distance between the mapped blobs is greater than a threshold to account for the maximum speed a bee could possibly travel. In these cases, the old trajectory index is dropped, and the newly detected bee is considered as the beginning of the new trajectory. A simple

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

Figure 3: Some images of bee #1 cropped by the image preprocessing program. Among these image, (a) and (b) were selected into the training set, while (c) and (d) were not.

greedy algorithm was implemented to realize this idea:

Input: $F_{1:n}$ as the 1st to n-th frame of the video, *threshold*
$i \leftarrow 1$;
$S_p \leftarrow detect(F_1)$; `returns a set of detected bee coordinations`
foreach bee x ***in*** S_p **do**
    $x.index \leftarrow i$;
    $i \leftarrow i + 1$;
end
for $a \leftarrow 2$ **to** n **do**
    $S_c \leftarrow detect(F_a)$;
    foreach bee x ***in*** S_c **do**
        $c \leftarrow \arg\min_y(\|x - y\|),\ y \in S_c$;
        if $\|x - c\| > threshold$ **then**
            $x.index \leftarrow i$;
            $i \leftarrow i + 1$;
        else
            $x.index \leftarrow c.index$;
        end
    end
    $S_c \leftarrow S_p$;
end

For each single bee trajectory, to smoothen its curve and denoise the measurements, the Kalman filter framework was implemented. Kalman filter is a hidden Markov model-based estimator which assumes linear models and Gaussian noise for the temporal dynamics of the hidden state the the measurement process. At each time frame, the hidden state estimation procedure of the Kalman filter can be described as follows [11]:

Time update Let (x_t, y_t) be the position of a bee at time t. The velocity vector of the bee, (v_{xt}, v_{yt}), was calculated as $v_{xt} = (x_t - x_{t-1}) \times FR$, $v_{yt} = (y_t - y_{t-1}) \times FR$, where FR was the frame rate of the camera, and (x_{t-1}, y_{t-1}) was its position at time $t-1$. The state vector of the bee at time t, X_t, was defined as $[\, x_t \; y_t \; v_{xt} \; v_{yt} \,]^T$. The *a priori* belief of its position at time $t+1$ is

$$X^p_{t+1} = AX_t,$$

where A is the time update model,

$$A = \begin{bmatrix} 1 & 0 & \frac{1}{FR} & 0 \\ 0 & 1 & 0 & \frac{1}{FR} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given the update model A, the covariance matrix P of the stochastic process X is updated by

$$P^p = APA^T,$$

where P^p denotes the updated covariance matrix.

Observation update The Kalman gain is a factor balancing the *a priori* belief and the observation, Y. Given the observation noise variance R, the Kalman gain is calculated as

$$K = PH^T(HPH^T + R)^{-1},$$

where H is the observation model that maps the state vector space to the measurement vector space. In our case H is an identity matrix. The *a posteriori* belief can be updated as

$$X_{t+1} = X^p_{t+1} + K(Y - HX^p_{t+1}).$$

P is updated by

$$P = (I - KH)P^p.$$

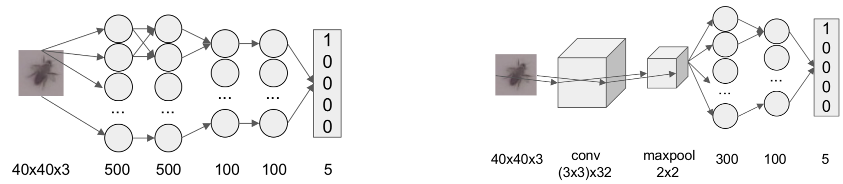
idTracker and its applications to bee tracking

The idTracker was introduced by Perez-Escudero *et. al.* in 2014 as a method for identifying unmarked animals [7]. Its idea is to compare the fingerprint of images in the training set of each individual to the fingerprint of the image to be identified. The fingerprint of an image is a modified color correlogram. To obtain this correlogram, the algorithm compares every single pair of the pixels in the image and counted the numbers of each $(d, i_1 + i_2)$ and each $(d, |i_1 - i_2|)$ calculated, where d is the distance between the two pixels, and i_1 and i_2 were the intensities of the two pixels. By comparing the color correlograms of the images to be identified and those of the reference images in the training set and looking for the most similar, the idTracker decides the identity of the individuals in the image. To compare two color correlograms, we subtract one from another and sum up the absolute values of the differences. However, calculating the color correlogram of an image was computationally expensive, with a time complexity $O(n^4)$, and it is not efficient to run the program in Python. Therefore, to expedite the computations, a C code was written and used for this purpose.

Individual identification with artificial neural networks

In this study, we applied an artificial neural network with only dense layers and a convolutional neural network to the bee identity classification problem. The architectures of the artificial neural networks used are shown in Figure 4. The architecture of the deep neural network was as follows: since the input was a 40×40 color picture, the input layer consisted of $40 \times 40 \times 3 = 4800$ units; the hidden layers were two 500 unit layers and three 100 unit layers; all activation functions between each hidden layer were rectified linear unit (ReLU) functions; the output layer used a "one-vs-rest" classification strategy and consisted of five units corresponding to the five recorded individuals. The activation function of the last layer was a sigmoid function.

The architecture of the convolutional neural network used in this study was as follows: for input, each image was 40×40 in size containing three channels. The first layer was a convolutional layer which had 32 filters with kernel size 3×3 followed by a max pooling layer with filter size 2×2. A same padding assured that the dimensionality was kept after the convolution was applied. After the first layer, the units were flattened and two fully connected layers of 300 and 100 units were applied.



(a) Neural Network with only dense layers (b) Convolutional Neural Network

Figure 4: The architectures of the artificial neural networks used in this study

All activation functions were ReLU functions. The output layer consisted of five units and was the same as in the neural network described earlier. All neural networks were written and trained in keras, an open source neural network library [12].

3 Results and Discussions

Video processing and training sets construction

Two kinds of videos were collected in this study: the videos with only one bee in the frame, which were used to generate the training sets and evaluate identification performance, and the videos with multiple bees in the frame, which were used to test to the tracking accuracy and individual identification in a mixing setting. All videos were first analyzed by the preprocessing program to recognize bees in the frame. For both kinds of videos, the overall accuracy of recognition was relatively high, which means that for most of the time, all bees in the video frame could be recognized by the blob detector. Nonetheless, since manually marking multiple bees in hours of video samples was quite onerous, there was no human labeled data generated in this study to be compared to, so there was no precise numerical evaluation of the preprocessing program.

The most common false negative situations were when a bee crawled too close to the edges of the metal meshes or next to a flower. The shadow of the meshes and the flowers changed the lighting on the bee and the surrounding, therefore made the blob less obvious. The most common false positive recognitions were the holes at the centers of the artificial flowers since they also looked like black blobs in the frame. However, we assumed that they can be easily recognized and removed in the data analysis step by looking for the detected blobs that do not move for the whole time. A different approach of distinguishing moving bees from the flower holes is to refer to the motion detector. With its help, the bee detector had a very low false positive rate. However, this approach did not work for resting bees.

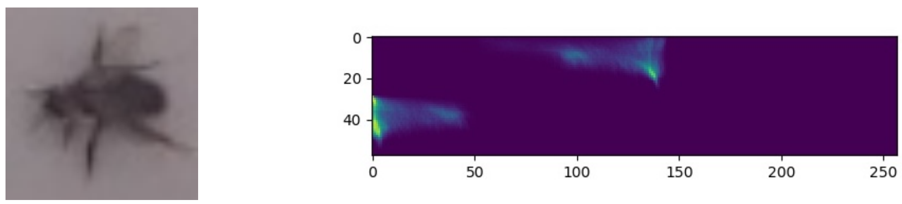
The training sets for five different bee individuals were generated from the videos of each of its own. The images in the training set were produced out automatically by the preprocessing program. Some low quality images were cropped by the program when a bee flew too fast to be captured by the camera, or when a bee hung on the meshes on the edges. These images, along with some repetitive images, were removed from the training set manually, and the final training set sizes for the five bees were 1011, 884, 980, 1067, and 1084, respectively..

Trajectory tracing and Kalman filter

The algorithm to match detected bees between two consecutive frames proposed previously was tested on the videos with five bees recorded at once. In the best case scenario, where the program never lost track of any bee, each bee got the same index over time. Otherwise, the program would end an index when a blob was lost and start a new index on any new blob detected. Even if the detection was lost for only one frame, the program would still start a new index, given the algorithm could only memorize the detection information from the last frame. Losing a detection in just one frame may not be noticeable to a human watching the video, but it was very likely in the video processing. In a test case on a one minute video, 380 new indices were assigned for five bees, which means about 4.8% of detections were lost due to various reasons. However, this algorithm had a very stable performance among different videos and it was reliable when the bees moved slower.

The Kalman filter framework was tested on a one bee video. When the Kalman filter was tested, the blob detector simply looked for the largest blob in the frame. Nonetheless, the program could still lose detection in some frames, mostly because the bee moved too fast and the camera could not capture. In this case, we increased the observation noise variance term R in the Kalman filter so it would rely more on the time update prediction and wait for the next detection. By comparing the raw data and Kalman filtered data in the video, we could visually inspect that the trajectory had been successfully traced by the Kalman filter, even if several detection were lost. A systematic quantification was not possible at this point since trajectory extraction by a human observer to be compared to was not performed.

Individual identification with idTracker



(a) (b)

Figure 5: An image of a bee and its color correlogram

The training sets we constructed previously on the five individual bees were used to test the performance of the idTracker. A sample color correlogram generated in idTracker is shown in Figure 5. The idTracker was tested on identifying the bees in the one bee videos. The confusion matrix was as follows:

$$\begin{bmatrix} 0.973 & 0.007 & 0.020 & 0 & 0 \\ 0.465 & 0.077 & 0.323 & 0 & 0.135 \\ 0.087 & 0.005 & 0.628 & 0 & 0.281 \\ 0.277 & 0.012 & 0.518 & 0 & 0.193 \\ 0.146 & 0 & 0 & 0 & 0.854 \end{bmatrix}$$

and the classification accuracy was about 50.6%.
The most prominent issue was that the classifier was biased and tended to misclassify other bees as bee #1, bee #3 and bee #5. The deep neural network model had also been tried on learning the color correlograms in the idTracker, but it turned out to be not feasible. When the data were plugged into the neural network, the training accuracy plateaued after the second or third round of epoch at about an accuracy of 60% and no longer improved as the training continued. Either the algorithm got stuck at a local minimum, or this feature space was inadequate as it did not carry enough information for the classification of the bees, which is consistent with the results presented in the confusion matrix above.

Individual identification with artificial neural networks

The artificial neural network with only dense layers and convolutional neural network model were implemented directly to the cropped bee images training sets. After 1500 epochs of training, the deep neural network had a cross-validation set accuracy of about 97%, and the convolutional neural network 98%. They were also incorporated with the video preprocessing program and tested on the one bee videos. The confusion matrix for the deep neural network was

$$\begin{bmatrix} 0.875 & 0.002 & 0.071 & 0.051 & 0 \\ 0.338 & 0.007 & 0.524 & 0.117 & 0.014 \\ 0.010 & 0.001 & 0.946 & 0.042 & 0.001 \\ 0.009 & 0.001 & 0.168 & 0.822 & 0 \\ 0.024 & 0 & 0.744 & 0.142 & 0.091 \end{bmatrix}$$

resulting in an average accuracy of 54.8%. The confusion matrix for the convolutional neural network was

$$\begin{bmatrix} 0.686 & 0.006 & 0.019 & 0.184 & 0.105 \\ 0.030 & 0.189 & 0.462 & 0.121 & 0.197 \\ 0.031 & 0.067 & 0.830 & 0.042 & 0.031 \\ 0.035 & 0.065 & 0.020 & 0.875 & 0.004 \\ 0.005 & 0.002 & 0.125 & 0.248 & 0.621 \end{bmatrix}$$

Giving an accuracy of 64.0%. Similar to the idTracker, they also suffered from the bias issue and misclassified other bees as bee #1, bee #3 and bee #5. Additionally, although we used the "one-vs-rest" strategy, sometime multiple units in the output layer would be activated. The discrepancy in the accuracies of identifying the cross-validation sets versus identifying bee in the video was noticeable. One possible reason for this discrepancy might be that the training set had been "cleaned up" manually, so some cases in the videos were represented in the training set.

4 Conclusions and future directions

In this study, we proposed a framework of tracking unmarked insects in a confined and controlled experimental setting. Given very affordable equipment, such as a webcam and a personal computer, we were able to track multiple insects steadily even when their trajectories overlapped and interrupted. We demonstrated the feasibility of this framework by developing and testing programs for each component of the framework on bumblebees, including video preprocessing, bee detection, trajectory tracing, Kalman filter for measurement denoising, the idTracker classifier, deep neural network, and the convolutional neural network.

In our testing, the video bee detection and the Kalman filter had the expected performances, while the others all have room for improvement. For trajectory tracing, the current program only offers the memory for one previous frame, so it could not deal well with the situations where two or more consecutive frames are lost. The identification programs, including the idTracker and the neural networks, all suffered from being biased on one or two individuals, which may indicate that the training set itself might be biased. To test this hypothesis, a different training set on the same bees as well as new training sets on other bee individuals can be generated and tested given more time. Although a mass amount of research have been done on supervised machine learning, based on various dataset and data quality, the fine tuning of a machine learning program mostly still remain a case-by-case trial-and-error process [13]. Due to time constraint, we haven't had the chance to pursue the ideas described above or further optimize the classifiers in this study.

The vision of this study is to test and compare different algorithms and investigate their feasibility of tracking bumblebees in the framework we proposed in a foraging setting. There are some parts of the framework that are not yet included in this study, such as associating all the trajectories of one identical bee together in one file to generate the final output of the tracking. Also, in the current stage, the programs still need some human assistance on certain steps, such as cleaning up the training set, tuning some parameters in video preprocessing, etc, although given a controlled, unchanging experimental setting, some of the human assistance only need to happen once. Continuing working on this framework, we hope to develop a more automatic procedure, and finally present this framework as a reliable and integrated software package available for the community by publishing it on GitHub.

References

[1] Nordus, L.P.J.J., Spink, A.J., Tegelenbosch, R.A.J., *EthoVision: A versatile video tracking system for automation of behavioral experiments*, Behavior Research Methods, Instruments, & Computers, 2001, 33:3, 398-414

[2] Risse, B., Mangan M., Del Pero, L., Webb, B., *Visual Tracking of Small Animals in Cluttered Natural Environments Using a Freely Moving Camera*, 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017. doi:10.1109/iccvw.2017.335.

[3] Straw, A.D., Branson, K., Neumann, T.R., Dickinson, M.H., *Multi-camera real-time three dimensional tracking of multiple flying animals*, Journal of the Royal Society Interface, 2011, 8, 395-409. doi:10.1098/rsif.2010.0230

[4] Branson, K. *Distinguishing Seemingly Indistinguishable Animals with Computer Vision*, Nature Methods, 2014, 11:7, 721-722. doi:10.1038/nmeth.3004.

[5] Dell, A.I. et al., *Automated image-based tracking and its application in ecology*, Trends in Ecology & Evolution, 2014, 29:7, 417-428. doi:10.1016/j.tree.2014.05.004.

[6] Rossetti B.J., Dynes T., Brosi B., de Roode J.C., Kong J. *GRAPHITE: A graphical environment for scalable in situ video tracking of moving insects*, Methods in Ecology and Evolution, 2017, 00:1-9. doi:10.1111/2041-210X.12944

[7] Perez-Escudero, A., Vicente-Page, J., Hinz, R.C., Arganda, S., de Polavieja, G.G., *idTracker: tracking individuals in a group by automatic identification of unmarked animals*, Nature Methods, 2014, 11, 743-748. doi:10.1038/nmeth.2994

[8] Dreiseitl, S., Ohno-Machado, L. *Logistic regression and artificial neural network classification models: a methodology review*, Journal of Biomedical Informatics, 2002, 35:5-6, 352-359. doi:10.1016/S1532-0464(03)00034-0.

[9] LeCun, Y., Bengio, Y., Hinton, G., *Deep learning*, Nature, 2015, 521, 436-444. doi:10.1038/nature14539

[10] Goodfellow I., Bengio Y., Courville A., *Deep Learning*, MIT Press, 2016. http://www.deeplearningbook.org

[11] Kaipio J.P., Somersalo E., *Statistical and Computational Inverse Problems*, Springer, 2015, ISBN 0-387-22073-9

[12] *Keras Documentation*, https://keras.io/

[13] Ng, A., *Deep Learning Specialization: Master Deep Learning, and Break into AI*. https://www.coursera.org/specializations/deep-learning