Marc Roizman

Fall 2019 Research: Using Machine Learning to Track Yellowjacket Flight Patterns

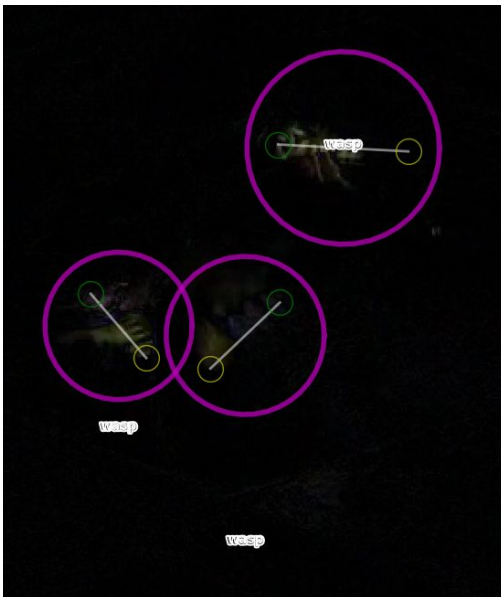
By Marc Roizman, Advised by Professor John Ringland

Yellowjackets are incredibly prevalent during North American summers, and you're bound to find a few if you go outside and keep an eye on the ground. But while much has been written about yellowjackets' venom, social habits, and nesting, there is a notable lack of academic research into yellowjacket flight patterns. After observing how yellowjackets hover around surfaces and objects, almost as if performing a sort of spacial detection, we decided to try to analyze some data.

To start the project, we decided to use a mix of footage recorded by Professor Ringland, which includes a mirror in it so as to allow us to identify the yellowjackets' full 3-dimensional positioning and directional orientation. We also found a very fascinating YouTube video recorded by user Oakheart, entitled "Messing with Yellow Jacket Wasp Nest - Flight Patterns". In their video, a branch is dropped onto a yellowjacket nest, and dozens of them swarm out to inspect the new object.

Our goal is to create a Neural Network which is able to locate and determine the orientation of yellowjackets with very good precision. In order to do this, we need a large training set. Luckily we have several long video clips of yellowjackets with a still camera. I wrote a Python script using the CV2 library which allowed me to subtract the average frame from any given frame of a video, thus allowing me to generate images where the yellowjackets appear as light colored shapes on an otherwise black background. With these images, I was able to use Professor Ringland's program to tag yellowjackets. We decided to tag a point on the front (between the antennas) and a point on the very back of every yellowjacket. This is a work in progress, and I currently

have around 500 yellowjackets tagged and identified from different frames of our selected videos.



A diff of a frame from the YouTube video with several yellowjackets tagged. The difficulty of obtaining precise pixel coordinates for both sides of the yellowjacket in blurry frames is something which we have to consider in the development of the training set.

We have also put some work into setting up the Convolutional Neural Network itself to detect yellowjackets in images. For this, we based the neural network design off of a tutorial on PyImageSearch.com. My code makes use of Keras RetinaNet and TensorFlow as well.

To test the CNN, I set up a very large training set of 2000 test images. To simplify this process, I created them using Inkscape. The benefit of creating the images is that rather than tagging, I can automatically obtain accurate information on yellowjacket position and directional orientation, since I need that information to draw the image.

A sample of one of my automatically generated yellowjacket images which I'm using to train the neural network.

The drawing process involved taking a large (1920x1080p) background, rotating it, and grabbing a random 256x256 square of it. Since the background is fairly varied, it simulates a random outdoor nature background pretty well. After doing this, I add a transparent 32x32 photo of a yellowjacket in. The yellowjacket is first rotated a random amount of degrees, then is translated a random number of pixels in both the x and y coordinates, and then finally has its RGB color matrix changed. This has the effect of helping the

CNN to better learn how to deal with a wider variety of scenarios for recognizing yellowjackets. The more differences between each image, the more likely the neural network is to properly learn how to identify *any* yellowjacket, as opposed to just this specific one.

As of the end of the Fall 2019 semester, I'm up to testing the CNN. My biggest test yet involved feeding it the 2,000 random yellowjacket images as an input, along with the x, y, sin(t), and cos(t) values for each yellowjacket. We opted to use sin(t) and cos(t) instead of just the angle, because it keeps the numbers continuous (Rotate something near 2pi degrees just a little bit, and you'll end up back at 0, but the cosine and sine will barely change.

Unfortunately the CNN was unable to predict anything on the set of 32x32 yellowjacket images, even after a 2,000 image, 200 epoch training run of the CNN. We will have to decide how to proceed since it seems that trying to locate and precisely determine all 4 parameter values for such a small part of the image might be too hard of a problem for a custom made model. What I may end up working to build is a hybrid model that uses an already existing object detection model to zero in on the yellowjacket, and then uses my own custom model to determine its (x, y, sin(t), cos(t)) parameters from the zoomed in image.

A sample of one of the "zoomed-in" yellowjacket images. The yellowjacket is scaled up to 200x200.

To verify the custom CNN could handle a zoomed in image, I tested the same network on pictures with the yellowjackets blown up to 200x200 (I also removed the color-changing to simplify the learning). This test was, like the previous one, a 2,000 image, 200 epoch training run. It was much more

successful, as shown by the plots below. It's likely that a longer test would've had even more success, as the scatterplots keep converging toward the correct values.

Below, I plotted the test values vs their predicted values (this was a test sample of 500 images—25% of the total). I plotted the x-location, y-location, and sin(t) & cos(t) for the rotation of t. The reason everything is between 0 and 1 is because all of my numbers are normalized, since CNNs are easier to train on normalized data...A sin(t) value of -1 would appear as a 0 on this plot.