Stephanie Huang
May 4, 2022
Professor Wloka
CS 153
Final Report

# Locust Orientation Detection using Classical Computer Vision Techniques

Project repository: https://github.com/StephanieH20/CS-153-Project

In this project, we use classical computer vision techniques to detect real-time locust orientation from video footage. From the Python OpenCV library, we used a combination of contour detection, Principal Component Analysis (PCA), and some experimental motion tracking techniques to help detect the orientations of multiple moving locusts. We found that this method produces fairly accurate results but it is not very resilient to noise.

## Motivation:

Locusts are a species of grasshoppers known for their swarming behavior. In large numbers, these insects breed rapidly and are capable of devouring entire crops, posing a major threat to agriculture [1]. Much of the locusts' success is owed to their well coordinated movements within swarms. An individual's movement is influenced by its neighbors, forming a collective mindset that allows the swarm to optimize their movements [2]. Researchers have been studying the movements of locusts in smaller groups in order to better understand their behavior.

Professor Weinburd and his research team at Harvey Mudd College have been trying to develop dynamical models as a way to predict swarm behavior. Specifically, they've been investigating hopper bands, which are coordinated groups formed by juvenile locusts that hop across the ground. The data for their research is primarily video footage. Prof. Weinburd's team were able to track the positions of locusts from these videos. The locusts are tracked using automated video tracking and their positions are analyzed in relation to each other [2]. However, the team has not been able to detect the locusts' orientations, which they believe will provide some more valuable insights. In this project, we use classical computer vision techniques in order to achieve this.

Data:

For this project, we were provided with video data from Prof. Weinburd and his lab. This includes 27 minute-long videos of locust footage as well as two 10-second-long sample videos.

For each video, we are provided with the data for each frame. This contains information for each locust present in the frame, including its $x$ and $y$ positions, state (0 = standing, 1 = walking, 2 = hopping, NaN = unknown), ID, and an estimated orientation angle.

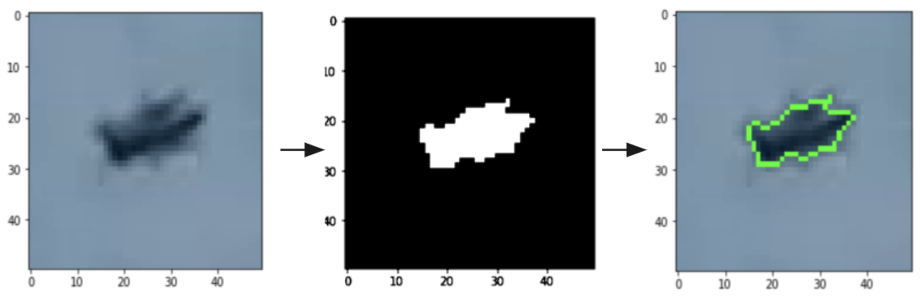
The given orientation angles were inferred by Prof. Weinburd based on the locusts' movements. It's noted that the estimations for locusts in the standing and hopping states aren't very accurate. Walking locusts have the most accurate estimates while standing ones have the least; their orientations are determined by the locust's prior motion. For hopping locusts, the accuracy is somewhere in-between. We will treat the inferred angles for walking locusts as ground truth and give less weight to those in other states.

The data was provided in .mat files, which we extract so they can be accessible for our Python script.

Methods:

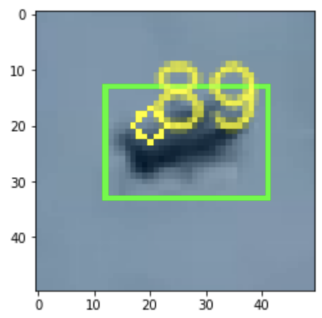
Our pipeline consisted of two major components: locust tracking, and orientation detection.

To track the locusts, we first applied contour detection using tools from OpenCV. For each frame, we apply a binary mask, which isolates possible locust regions from the background, and then apply OpenCV's contour detection algorithm on the resultant:

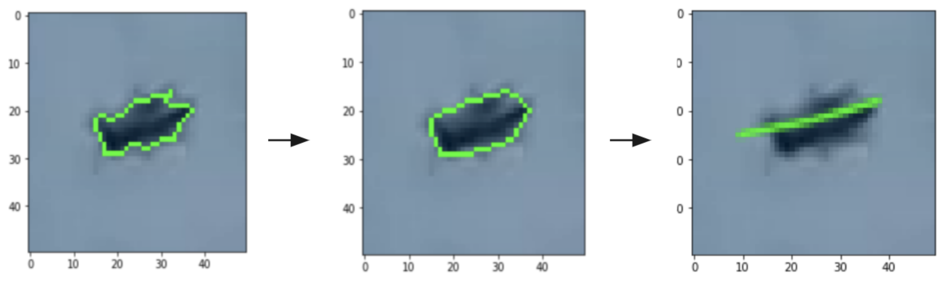


The next stage is to identify and track the locusts over the course of the video. Using the given positions of the locusts from the data, we were able to match any potential locust contours to their corresponding IDs. This was done by checking which positions were near or contained within which the contours. Additionally, this would also remove any extraneous contour detections.

The given positions weren't always aligned with the locust's center. A few positions would be close to the locust but entirely off the body. So for better localization, we checked if the positions were contained within a contour's maximal bounding box.



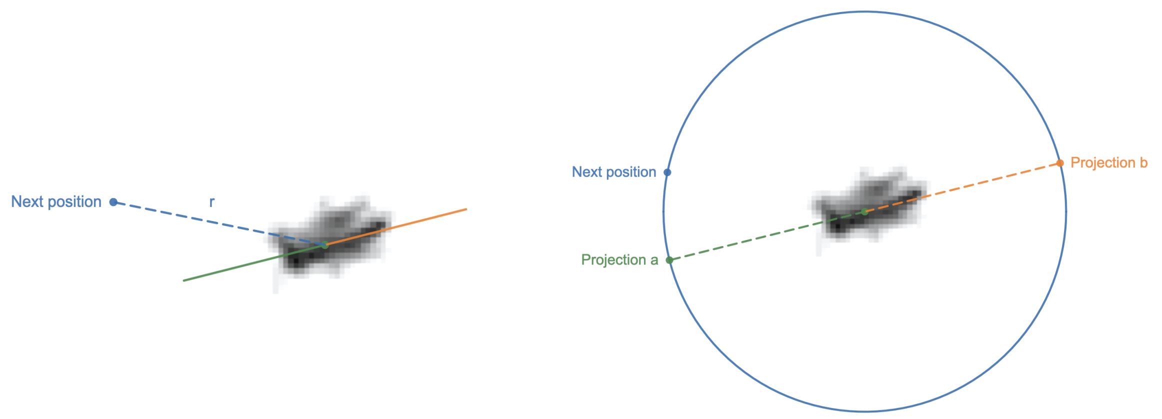
To detect orientation, we assumed the direction of longest length corresponded to the orientation axis. Principal Component Analysis [3], was then performed on the convex hull of each contour to determine this axis:



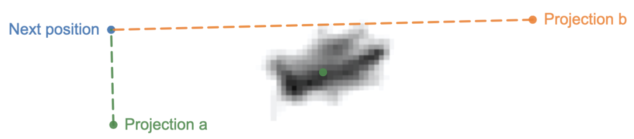
The last major step was to determine the locusts' direction of orientation (facing direction). While we have the axis of orientation, we have yet to compute its direction. A locust's direction of motion is almost entirely determined by its orientation [2], so the desirable angle should accurately estimate its next position.

Out of the two angles given by the axis, we choose the angle that most accurately predicts the locust's next position.

The length, $r$, between the locust's current position and next position is calculated. We then project the locust's movement over range $r$ for both angles:



We choose the angle that minimizes the distance from the next position:



The error is denoted as the minimum angular difference between the calculated angle and the actual angle, as given by:

$$error_{f,i} = \left(100\left|calc_{f,i} - actual_{f,i}\right| \bmod (200\pi)\right) \quad \text{for locust } i \text{ at frame } f.$$

(The angles are scaled by 100 to deal with modular operations as they are cyclic)

For accuracy, the error is converted into a percentage, denoted by:

$$accuracy_{f,i} = \left(1 - \frac{\min\left(error_{f,i},\, 200\pi - error_{f,i}\right)}{100\pi}\right)$$

Each accuracy rating is scaled by a weight, $w$, depending on the current state of the locust. As noted earlier, the given angles for walking locusts tend to be the most accurate, while the given angles for stationary ones were the least, and those for hopping were somewhere in-between.

The weights of stationary, walking, and hopping locusts were 0.5, 1, and 0.75 respectively.

Net accuracy is calculated as:

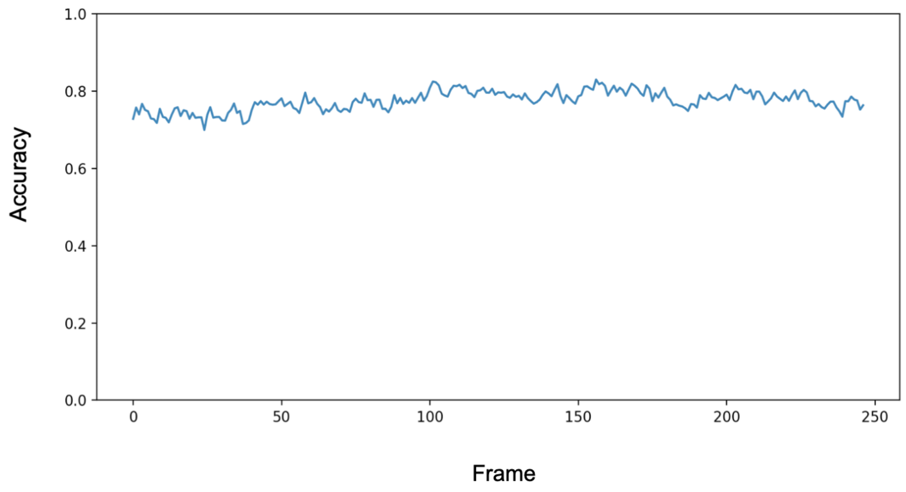
$$\frac{\sum_{f \in frames,\, i \in locusts} w_{f,i} \cdot accuracy_{f,i}}{\sum_{f \in frames,\, i \in locusts} w_{f,i}}$$

<u>Results:</u>

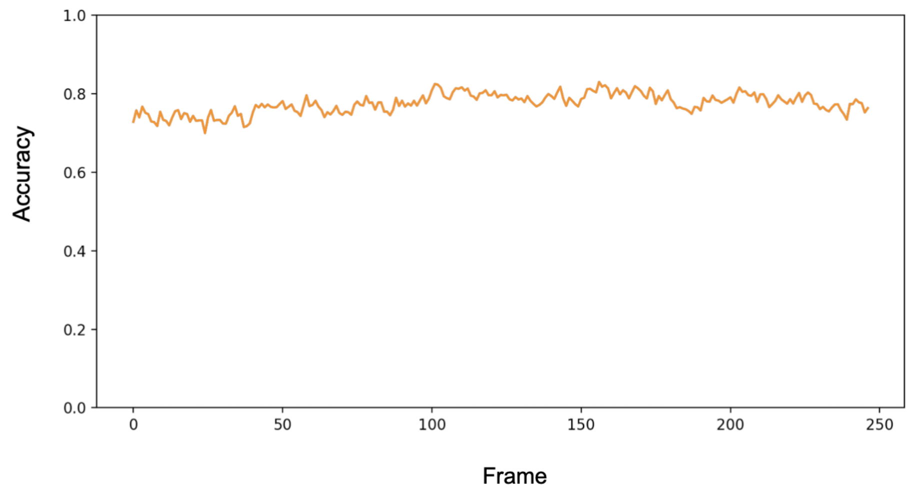
Running the algorithm on the 10-second sample videos `preprocessed_133_10sec_710.avi` and `preprocessed_133_10sec_1910.avi`, the accuracy in detecting locust orientations were 77.75% and 74.84% respectively, averaging to around 76.30%.

Tracking the accuracy, this seems to be somewhat consistent across all frames:

Video `710.avi`:



Video `1910.avi`:

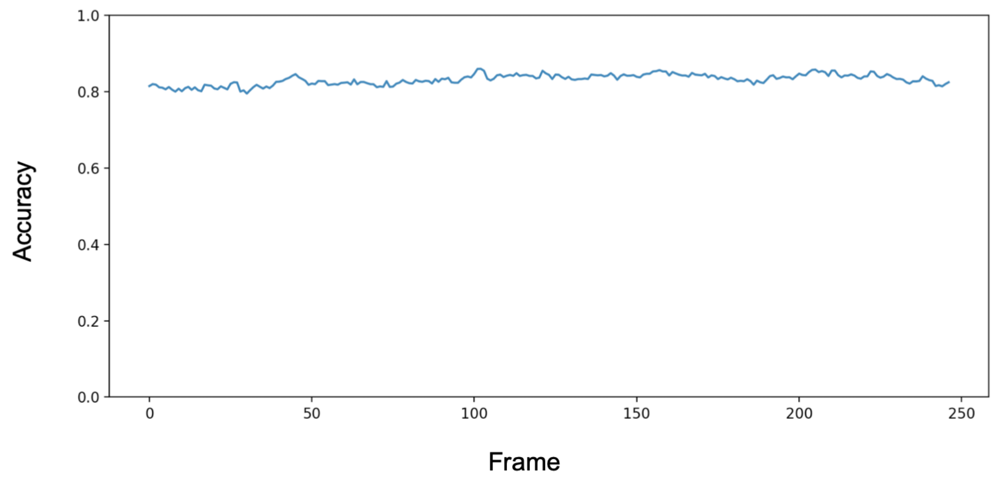


The accuracy metric was a bit harsh when the algorithm determined the correct axis of orientation but incorrect direction. Accuracy depends on the minimum angular difference between angles, so a detected angle in the opposite direction would have a 0% accuracy, despite being along the same axis.

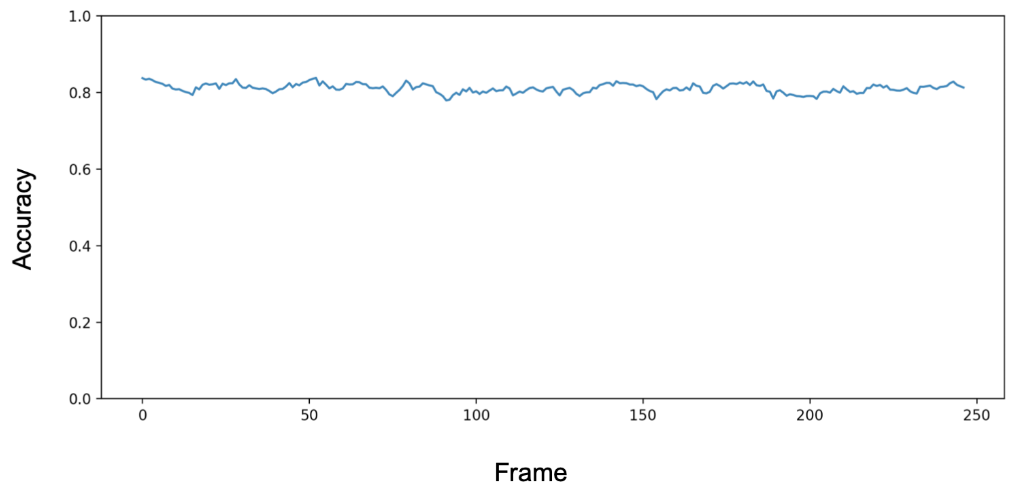
To investigate this further, we observed the accuracy of the axis detection and found it was somewhat higher.
For `preprocessed_133_10sec_710.avi` and `preprocessed_133_10sec_1910.avi`, the accuracy in detecting locust axes were 81.06% and 83.30% respectively, averaging to around 82.18%.

The accuracy was also more consistent across all frames:

Video `710.avi`:



Video `1910.avi`:



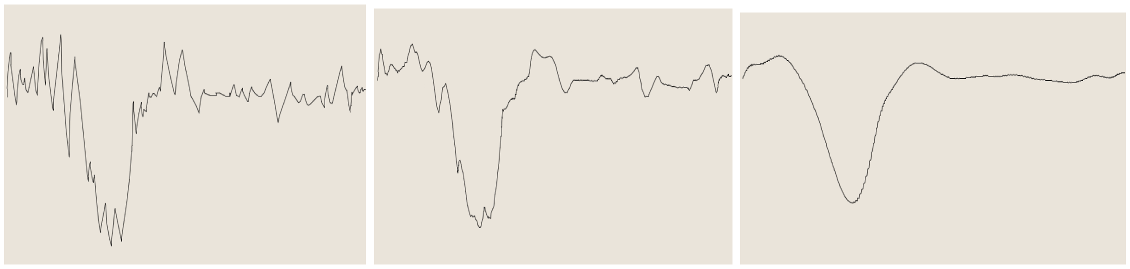
<u>Analysis and further exploration:</u>

As seen in the data trends, some of the error results from inaccurate direction detection. While the algorithm may accurately determine the axis of orientation, if it incorrectly determines the direction, it can throw the accuracy toward the other extreme: it's near 100% if the correct direction is chosen but near 0% otherwise.

The sudden direction switches are an issue for the algorithm. Our direction detection method relies on the locusts' motion, so it may be inaccurate for standing locusts. A possible explanation could be that tiny changes or noise in the positions of stationary locusts are detected and interpreted as legitimate movements. A simple fix would be to apply motion-based detection only when locusts are moving while standing locusts' orientation would be unchanged. But this would also create additional problems if locusts are turning in place as they'd still be stationary but their orientation would change.

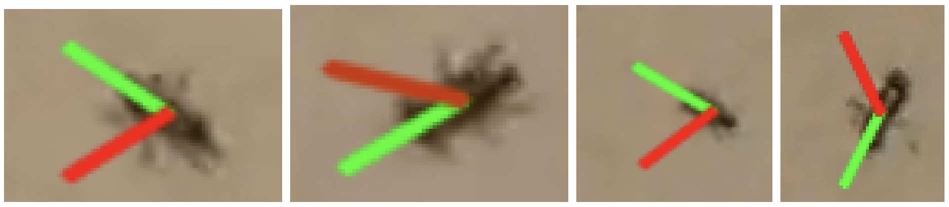
For further exploration, I considered using curve shortening to deal with any noise in the locusts' paths. This method can be used to deal with noise within static images by gradually reducing image value curvature [4].

[5]



Applied temporally, we can reduce any noise within a locust's trajectory. While this does not preserve the positions, it can help deal with the mentioned problems we encountered in angle detection.

Another explanation for the error could be inaccurate data. We used the provided angles for walking locusts as ground truth, but in some instances, they can be fairly inaccurate:



Given orientations are in red, calculated orientations are in green. It seems that our calculated orientations give a better representation of the locusts' actual orientation, rendering the error inaccurate.

Bibliography

[1] G. Ariel and A. Ayali. “Locust Collective Motion and Its Modeling”, 2015. https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004522#sec001.

[2] J. Weinburd, J. Landsberg, A. Kravtsova, S. Lam, T. Sharma, S. Simpson, G. Sword, and J. Buhl. “Anisotropic Interaction and Motion States of Locusts in a Hopper Band”, 2021. https://www.biorxiv.org/content/10.1101/2021.10.29.466390v3.

[3] *Introduction to Principal Component Analysis (PCA)*. OpenCV. (n.d.). Retrieved April 14, 2022, from https://docs.opencv.org/3.4/d1/dee/tutorial_introduction_to_pca.html

[4] M. Bertalmío and S. Levine. “Denoising an Image by Denoising its Curvature Image”, 2012. https://www.ima.umn.edu/sites/default/files/2411.pdf