Grace M. Rodríguez Gómez
802-09-7188
CCOM44995-002

LabelBee Research Project

Introduction:

The main goal for the Label Bee open source software is to develop a platform that analyses individual insect behavior and acquires new observations into the role of individual diverse behavior on the bee colony performance. It is meant for biologists to make annotations based on observations to joint videos, remote visualization and data acquisition that monitors thousands of marked bees over a extended and continuous period of time [4].

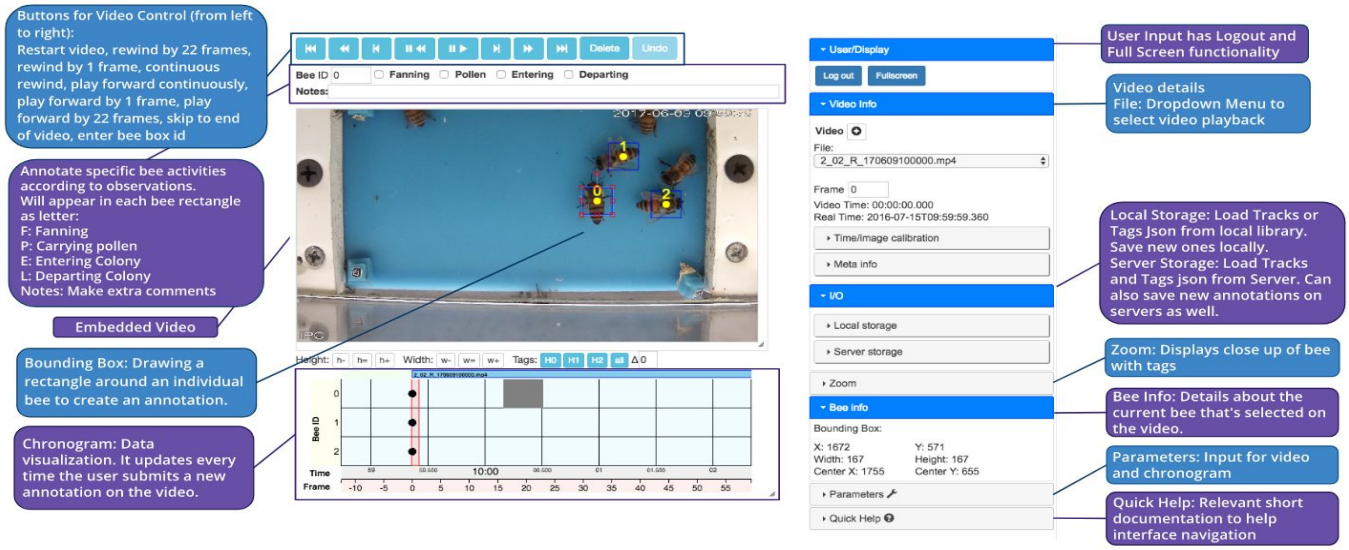
This project is a collective work between biologists, computer scientists students and professors from the University of Puerto Rico - Mayaguez and Rio Piedras Campus supported by NSF. My individual contribution was mostly to the project's visualization, the chronogram graph.

Technology Used:

1. D3: The Data-Driven Documents or D3, is an open source JavaScript library for manipulating documents based on data and create data visualization. It is a fast and flexible API that supports large data sets and dynamic behaviors and interaction[2].D3 works by allowing the user to bind data to a Document Object Model (DOM) and then apply data-driven transformations to create tables or interactive charts [5].
2. Json: JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange[3]. Its is meant to exchange and store data between the browser and the server.
3. Main programming language: Javascript

Interface:

The interface has the ability to create and record annotations in its embedded video by manually labeling bees. The main functionality of the interface is explain in the following illustration.



Chronogram:

A chronogram is an inscription, sentence, or phrase in which certain letters express a date or epoch. For this project, the chronogram records and displays a frame log for different honey bee behaviors. These behaviors or activities are classified as entering, exiting, fanning, pollen or none. The user has the option of marking an individual bee in the video and wherever it's doing an activity. The y axis of the chronogram

represents the bee's ID and the x axis of the chronogram represents the frame where that bee was annotated. There are two types of primitives in the chronogram, circles, which represents the bee that was annotated within one frame and rectangles, which represents the bees with the same ID that were annotated continuously within a plus one frame of difference.

The different bee activities are represented by color. Entering is represented by green, exiting is red, pollen is yellow and fanning is purple. If the annotation is just a circle, the circle changes color depending on the last activity marked. If the annotation is a rectangle, the activities are seen as smaller rectangles, where the entering rectangle is on the first x coordinate of the interval, the exit rectangle is on the last x coordinate of the rectangle, pollen is inside the interval but it's seen in the first half of the interval rectangle and fanning is on the second half of the interval rectangle.

If the user hovers in top of a gray rectangle or in top of a circle, they will see the details about that particular interval.

Architecture:

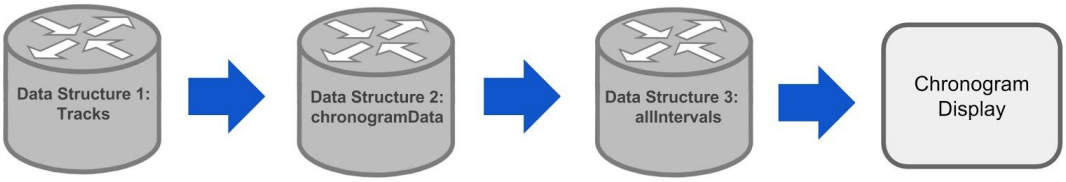
The data is managed through a Model-View-Controller design pattern. The Model encapsulates the data and defines the logic and computation that manipulates the data. The view is a visual representation of the model [1]. In the case for the Label Bee software, the view is all of the software's user interface (UI).

The input for the user is drawing a box around a bee on the embedded video. Once the info is submitted, that data is stored at the model to the Tracks data structure and then converted to chronogramData. The one that updates that data is the controller, which handles the input by binding the data in the form by different variables [5].

The data flow architecture for the project starts with the videos been recorded at the Experimental Station in Gurabo. Since the web interface is meant to give access to an abounding amount of high definition videos, this data will be stored and be accessible in the servers at the High Performance Computing Facility at Puerto Rico. The data is accessed from the servers from the controller, who is also receiving input from the user through the interface and updating the data in the data structures. After receiving updated data from the data structure, the controller also has the job of sending those updates to the view for display.

Internal Development:

The best way to explain the general implementation of the project is as a series of different data structures receiving specific data from each other. The first data structure that directly receives the input from the user in the video interface is called Tracks. Tracks is a Json file that can also be saved locally to be reused. The data from Tracks is passed to another data structure called ChronogramData, which saves in each element the x, y coordinates and activity of each annotation. From ChronogramData the data is then saved to a new data structure called allIntervals. allInteval is a list of objects where all the annotations are saved as intervals. Its attributes x1 is the first frame the bee was marked, x2 is the last frame the bee was marked, y is the bee id's and activity is the marked behavior. The following illustrations is the data structures diagram.



Once allIntervals is updated, this data will be filtered to objects that will be displayed as the svg primitives that represent0 the annotations. These objects can be expressed as the result of filtered data that flows to the functions that create and append the primitives(circles and rectangles) to the svg canvas of the chronogram. For each activity, there is a new object been made that calls the activity's own create and update functions. There is also another function and a new object for the plain intervals rectangles.

In order to display the interval rectangles and activities in the desired order in the chronogram, we have to add three new layers to the svg canvas. This will allow us to append each primitive to the layers in a sorted order. First, a bottom "g" element is created for the intervals rectangle which is gray. In top of that, a middle "g" layer is created for the visuals primitives of pollen and fanning. Afterwards, a top "g" layer is created for the last activities' primitives that are gonna be displayed, which are entering and exiting. The circles are added to the chronoGroup element, which is always going to be in top of the added layers.

How to use:

1) Go to the menu on the right of the embedded video and press I/O to load a new video. There is the option to add a new video or use the dropdown menu next to File that has videos that are available in the server. Usually, there will be a default video that loads on the web app automatically. In this case, it's the 2_02_R_170609100000.mp4.
2) Once the video loads, go inside the video and by pressing "Shift" and left mouse click in top of a bee, you make a rectangle around the bee with a new id.
3) If you marked more bees in the same frame, they will be automatically be assigned incrementing numbers starting from 0.
4) You can press the play button and keep marking bees. By now, the chronogram will display circles representing the bees if they were marked with more than one frame of difference of each other.
5) In order to display as them as intervals, mark the bees with only one frame of difference within each other. This can be easily achieved by pressing shift and right mouse click on the first bee, pressing forward button next to the play button and repeating this step until desired.
6) To marked the bees with activities, simply mark the bee and then check on the rectangle next to the activity depending on bee behavior seen in the video. In order to keep seeing the activities in the interval, they have to be constantly be checked when the annotations for the bee are being made.
7) As another note, both the embedded video and the chronogram canvas can resized to the prefered size. This is done by moving the mouse to the end of the borders of either one, pressing right mouse click and moving it to the left or right from the original size.
8) The chronogram canvas also has the ability to zoom in or out. For the chronogram, press Shift and mousewheel to zoom in/out.
9) The new annotated data can be saved as a Json or CSV by going to I/O, press Save Tracks for Json or Save Tracks(CSV) for CSV.
10) The user also has the option to load its own Track data by going to I/O, press Load Tracks and using the file on the local storage.

Bugs:

1) The 4px width exiting rectangle is been created about 1 px before the last coordinate of the rectangle. The implementation is the same as the entering activity, but it's not being displayed as it should.
2) Sometimes the details that are shown for each interval by hovering in top of the gray rectangles show what are supposed to be the details for the interval before.

References:

[1] Atwood, Jeff. "Understanding Model-View-Controller." *Coding Horror*. N.p., 05 May 2008. Web. 05 June 2017. <https://blog.codinghorror.com/understanding-model-view-controller/>.
[2] Bostock, Mike. *Data-Driven Documents*. N.p., n.d. Web. 05 June 2017. <https://d3js.org/>.
[3] "D3 API Reference." *D3*. Github, n.d. Web. 05 June 2017. <https://github.com/d3/d3/blob/master/API.md>.
[4] "Large-scale multi-parameter analysis of honeybee behavior in their natural habitat." *BigDBee Project*. N.p., n.d. Web. 05 June 2017. <http://bigdbee.hpcf.upr.edu/>.
[5] Serralles, Shirley, and Remi Megret. "Bee Tracker." (2016): n. pag. Web. 5 June 2017.