ApisFlow: a Real-Time Automated Tool to Detect, Classify and Count Honey Bees Castes at the Hive Entrance

Gabriel Vasconcelos Fruet[1], **Isac Gabriel Abrahão Bomfim**[3], **Rafael Capelo Domingues**[1], **Antonio Rafael Braga**[2], **Danielo G. Gomes**[1]

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Departamento de Engenharia de Teleinformática
Centro de Tecnologia, Universidade Federal do Ceará (UFC), Fortaleza - CE

[2]Redes de Computadores – Campus Quixadá,
Universidade Federal do Ceará (UFC), Quixadá-CE

[3]Laboratório de Apicultura, Campus Crateús,
Instituto Federal do Ceará (IFCE), Crateús-CE

[gabrielfruet,rafaelcapelo22]@alu.ufc.br

[rafaelbraga, danielo]@ufc.br, isac.bomfim@ifce.edu.br

Abstract. *There are three types of castes in honey bee (Apis mellifera L.) colony: the queen, workers, and drones. Although they are all important to perpetuate the species, drones do not collaborate with tasks in the colony and their counting may overestimate the real number of foraging workers, individuals who really contribute to pollination services. So, monitoring, classifying, and counting the flow and proportion of workers and drones through the beehive entrance provide useful information related to the colony's well-being. This has become possible thanks to the so-called Precision Beekeeeping, an emerging field of digital agriculture to gather and transfer bee-related data over time. Here, we propose ApisFlow, a real-time object-tracking framework for automatically detecting, tracking, classifying and counting the flow of honey bee castes at the hive entrance. ApisFlow uses computer vision and machine learning methods and algorithms. We strongly believe that ApisFlow allows bee counting, tracking, and classification in a less laborious, safe, fast, and accurate way to help beekeepers in making decisions saving time. Suggesting a high-precision algorithm with a mean error rate below 5%.*

1. Introduction

Precision beekeeping is an emerging field of digital agriculture that integrates Computer Science/Engineering, Biology, and Zootechnics to promote remote non-invasive monitoring of bee colonies. Thanks to digital infrastructures and computing methods, precision beekeeping allows monitoring honey bee (*Apis mellifera* L.) colonies seeking to support and improve pollination services, as well the well-being and production of these insects and beekeepers [Hadjur et al. 2022]. Although this bee species is the most used for pollination services in commercial plantations around the world [Khalifa et al. 2021], they need to meet some minimum standards in order to well perform their role as commercial pollinators. Therefore, some parameters can be used to verify whether and when a colony is suitable for this service [Delaplane et al. 2013]. Counting the flow of bees entering and

leaving the hive is a good indicator for that [Sagili et al. 2011]. Besides providing a good estimate of the colony strength, observing the flow of bees in the hive entrance associated with the detection and classification of the type of bee caste can provide more accurate information about the pollinator potential of a colony, as well as a prediction about swarming [Boes 2010]. Honeybee colonies have three types of castes: the queen, the workers, and the drones. Although they are all important for the perpetuation of this species on the planet, workers and drones are the castes that are most often seen entering and leaving the nest As drones do not collaborate with tasks in the colony and their numbers increase when the colony is heavily populous, their count may overestimate the real number of foraging workers, individuals who really contribute to pollination services. In addition, these drone counts can be used as swarming predictors, helping beekeepers not to have their colonies suddenly weakened, as long as they receive this information and quickly carry out adequate management to avoid this phenomenon [Boes 2010]. However, the presence of an observer in front of a beehive for counting and detecting bees, in addition to disturbing the activities of the bee colony, can be dangerous and exhausting for this observer, as well as it is not practical to be carried out continuously [Delaplane et al. 2013].

Therefore, to tackle this problem, computer vision algorithms based on Digital Image Processing (DIP) techniques can be used to monitor these variables at a distance and in real-time, facilitating diagnoses or predictions of certain events that are happening or will occur in certain colony or even in an entire apiary [Barros et al. 2021, Albuquerque et al. 2022, Andrijević et al. 2022]. From this perspective, here we propose ApisFlow, a framework capable of automatically detecting, tracking, classifying, and counting in real time the honey bee castes entering and leaving the beehive by methods and algorithms of computer vision and machine learning.

2. Material and Method

2.1. Videos dataset

We used a honey bee dataset from the Appalachian State University[1] to test and validate ApisFlow. This dataset has 1-minute long videos captured from the entrance of 24 beehives and has been recorded since April 2022, on a daily basis, every five minutes, from 7h:00 a.m. to 8h:00 p.m.

2.2. Hardware and Software

Machine learning and computer vision algorithms are typically CPU and GPU intensive as as they deal with billions of mathematical operations for making successful predictions. To tackle this challenge, we used a robust computer system that included a high-end GPU (GTX 1060 6GB), a sizeable 16GB of RAM, i5 7400, Windows 10, and GPU-powered algorithms. Additionally, we relied on Python 3.10.9, an advanced programming language, and leveraged a range of helpful libraries such as NumPy, SciPy, OpenCV, and the standard Python library.

2.3. ApisFlow flowchart

Figure 1 shows the ApisFlow dataflow. To use the ApisFlow framework, you need two things: (1) the videos you want to count bees, and (2) a trained object detector specific

[1] https://appmais.cs.appstate.edu/

to the dataset you are using with ApisFlow. Once you have these two requirements, you can start counting the data. The counting process happens in a real-time and online way, meaning it happens while the video is being processed, and each frame is only viewed once (3). ApisFlow detects objects in the images using the object detector you plugged in, then associates track, classifies, and counts them in that order (4). After processing each video, the counted data is saved (5) and exported for each class (6).

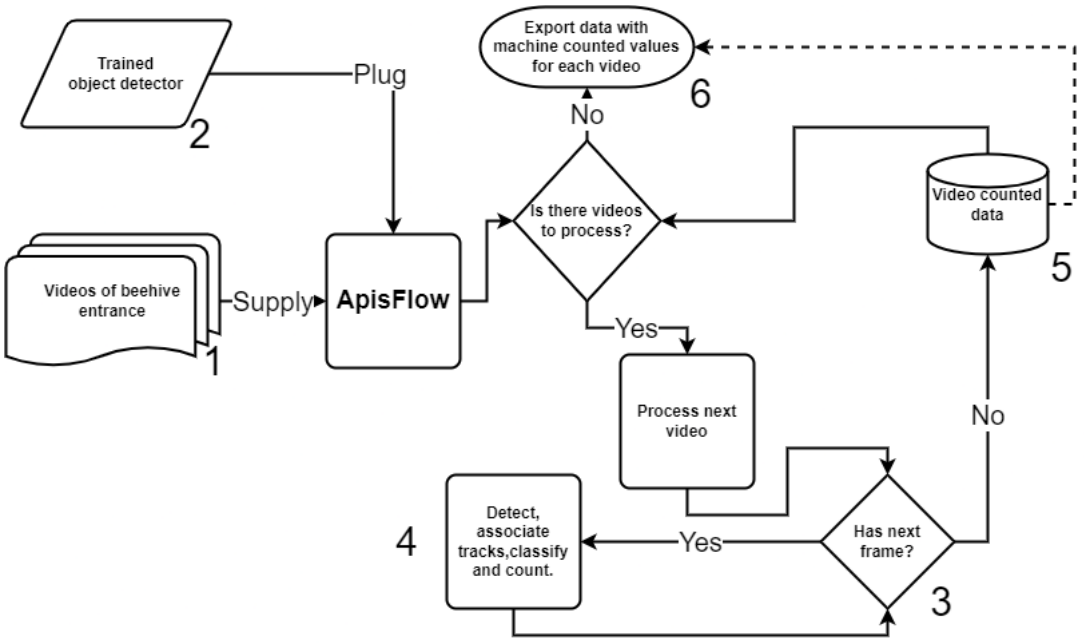


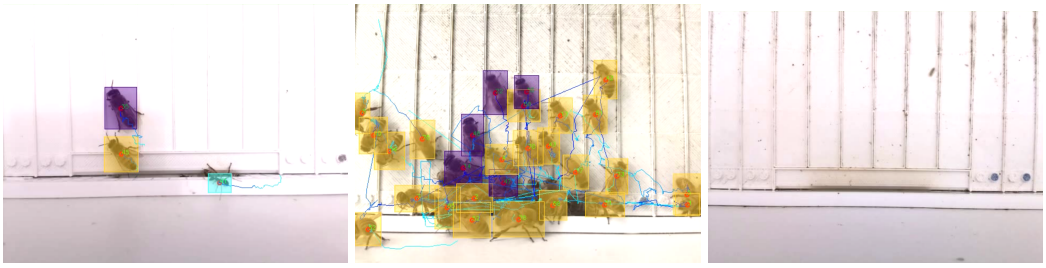
Figure 1. ApisFlow flowchart.

Over the last years, we have seen interesting advances in object detection tools such as YOLO[2] (You Only Look Once), *R-CNN* [Girshick et al. 2014], and *MobileNet* [Howard et al. 2017]. By following the best practices of software engineering, we defined interfaces for object detector creation and detection. This allows us to seamlessly use different algorithms without requiring modifications to the underlying code, only the object detector being used. In our tests, we mainly used YOLOv8, which is currently considered the most advanced real-time object detection method. YOLO is suitable for the purpose of this study because it achieves both high accuracy and fast prediction times [Yin et al. 2020].

2.4. Classification

The object detection algorithm plays a crucial role in determining the object's category within an image. In this study, we specifically focused on identifying and categorizing workers, drones, and other insects, referred to as potential enemies.

[2]https://docs.ultralytics.com/



(a) all classes (b) workers and drones (c) none

Figure 2. Classified objects*

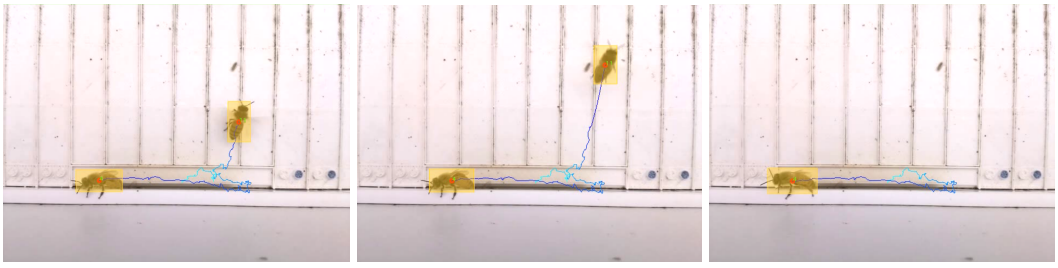
*A yellow box represents a worker bee; a purple box indicates a drone bee; a blue box represents a potential enemy. Additionally, the blue line depicts the path that a bee has traversed over time.

Figure 2 shows frames from videos processed by ApisFlow. Colored rectangles indicate the object's class: yellow for workers, purple for drones, and blue for potential enemies. Additionally, a blue gradient line represents the estimated path of the object.

During tracking, inconsistencies in object classification may arise as frames progress, particularly when dealing with objects that are very similar. To address this issue, we have empirically established a solution. For each tracked object, we store the 50 most recent classifications. If more than 20% of these classifications label the object as a drone or potential adversary, we assign it the most common class between the two. Otherwise, we classify it as a worker. We implemented this approach because training an object detector with precise differentiation between workers and drones is challenging. Given their similarities and the prevalence of workers in most images, the model occasionally misclassifies drones as workers. To mitigate this, we rely on this classification estimation method.

2.5. Estimating the bees movement via Kalman Filter

When monitoring honey bees at the entrance of the beehive, we observe their takeoff and landing activities. To detect honey bees, specifically foraging worker bees or drones, we employed object detection algorithms. However, these algorithms exhibit some uncertainty when determining the exact bounding boxes around the bees. To address this concern, we used Kalman Filter. The Kalman Filter is an iterative filtering process that helps refine the estimation of measured variables with a certain level of uncertainty, allowing us to obtain more accurate results.



(a) i (b) i+1 (c) i+2

Figure 3. Consecutive frames example.

Figure 3 shows three consecutive frames (*i, i+1, i+2*) of the outgoing and incoming flow of foragers and drones at the beehive entrance that were estimated through our Kalman Filter. For every object being tracked, we have a vector of dynamic states that describe the tracking estimated state throughout its existence. This dynamic state brings the 10 following features used in the Kalman filtering process: $x, y, a, h, v_x, v_y, v_a, v_h, a_x, a_y$, where x and y are the coordinates of the track in the frame, a is the aspect ratio of the bounding box that surround the object, and h the height of the bounding box, v means velocity and a means acceleration.

2.6. Object Tracking

Many improvements in object tracking have been discussed and implemented over the last few years. DeepSORT, for instance, is a simple, online and real-time tracking objects algorithm while assigning an ID to each object [Yang et al. 2022][3].

The main challenge of tracking is figuring out what news detection belongs to what existing tracks. To tackle this problem, we need to establish how probable is the combination of a track and a detection. To determine the likelihood of a track, we can make use of metrics to create a cost function for the combination of a tracked object and detection, that will be inversely proportional to the probability of being the same object.

A simple way to measure distance is by using the Euclidean distance, we use it to find the distance between where we expect something to be (*i.e* predicted state from Kalman Filter) and where we actually detect it. We also consider the difference in velocity between the predicted and detected positions in relation to the track. Another metric that we use is the cosine distance, this helps us see how much a path differs from what we expect, to calculate it, we look at the speed of the path and compare it to the speed of the detected object in relation to the path. IoU is another useful metric that we use to see how much two rectangles overlap, we use it by comparing the projected state bounding box and the detection bounding box.

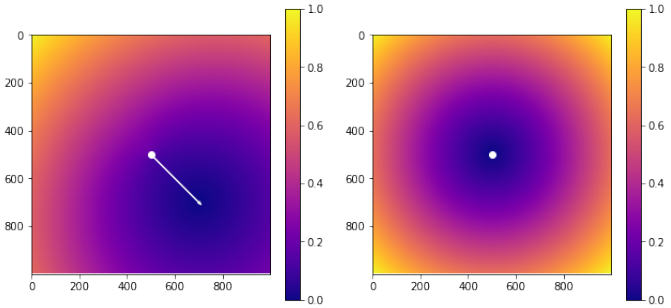
Based on these metrics, we can define the following equation for estimating a cost to a combination of a track and a detection:

$$||p_t - p_d|| \cdot \sqrt[3]{(\alpha + 1 - \frac{v_t \cdot v_d}{||v_t|| \cdot ||v_d||}) \cdot (\beta + ||v_t - v_d||)} \cdot (\gamma + 1 - IoU(b_t, b_d)) \cdot \lambda \quad (1)$$

where p is the position, v is the velocity and b is the bounding box, variables underscored with t are related to track, and with d are related to detection, greek letters are constants.

The user has the option to select the constants values and, in our tests, we used α = 0.5; β = 0.5, γ = 1, λ = 0.1. These constants determine the level of influence each metric has on the final score. Without these constants, certain metrics may suggest the optimal combination, while others may have disproportionate importance. Therefore, we apply a constant value to each metric to ensure a balanced evaluation.

[3]https://nanonets.com/blog/object-tracking-deepsort/



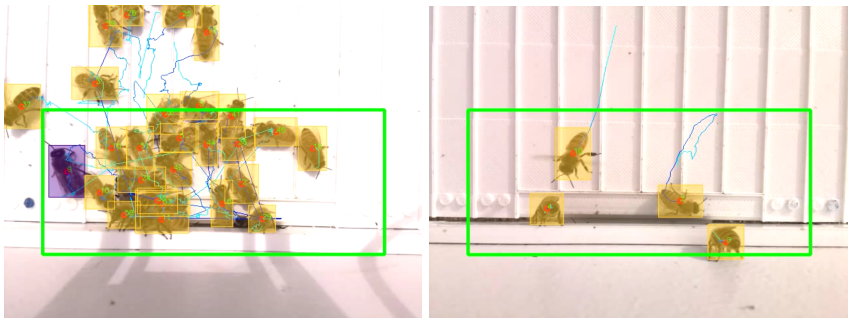
(a) non-null velocity (b) null velocity.

Figure 4. Bee flow heatmaps.

Figure 4 shows heatmaps of bee movement, where spots closer to 1 represent the highest cost and spots closer to 0 indicate the lowest cost. This implies that if detection occurs in a location with a low cost compared to a particular track, it is highly probable that it belongs to the same bee. The detection assigned to a specific track is determined by selecting the one with the lowest cost among all the detections, excluding the lowest cost associated with any other track. We solve this linear assignment problem using the Hungarian algorithm.

2.7. Counting

When measuring the flow of bees in a colony, the typical approach employed by beekeepers is to count incoming and outgoing bees based on their observable behavior. If a bee was outside and enters the beehive, it is considered an incoming bee. Conversely, if a bee was inside the colony and leaves, it is considered an outgoing bee. However, in the context of this paper, accurately determining whether a bee has truly entered or exited the beehive poses challenges. Bee entrance area is often congested with a multitude of bees, making tracking difficult and increasing the likelihood of confusion. To address this issue, we propose the use of a designated region surrounding the entrance, referred to as the ”Entrance Box” (Figure 5).



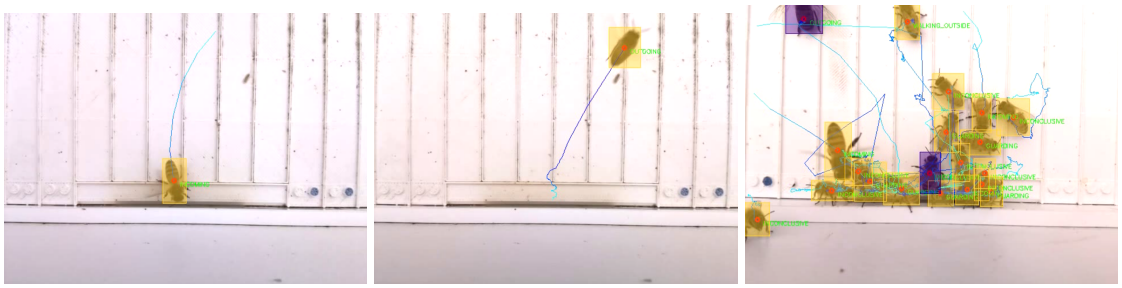
(a) congested bee hive entrance (b) entrance without congestion

Figure 5. Entrance bounding box (green rectangle).

Referring to the green rectangle shown in Figure 5, we can readily determine whether a bee is positioned inside or outside of it. Building upon this observation, we established specific states for each tracked bee by considering only their initial and current positions. These states are summarized in Table 1 and illustrated in Figure 6.

Table 1. Bees flow states.

Initial	Current	Flow State
Inside	Outside	Outgoing
Outside	Inside	Incoming
Inside	Inside	Guarding
Outside	Outside	Walking outside
Less than 5 frames tracked		Inconclusive



(a) incoming (b) outgoing (c) all states

Figure 6. Bees flow states.

2.8. Error measurement

During the development of the counting algorithm, it is essential to assess its effectiveness and accuracy in predicting bee counts. The simplest method we employed for measuring this involves manual counting (named here as real) of incoming and outgoing bees, which is then compared to the counts generated by the automated system (named here as measured).

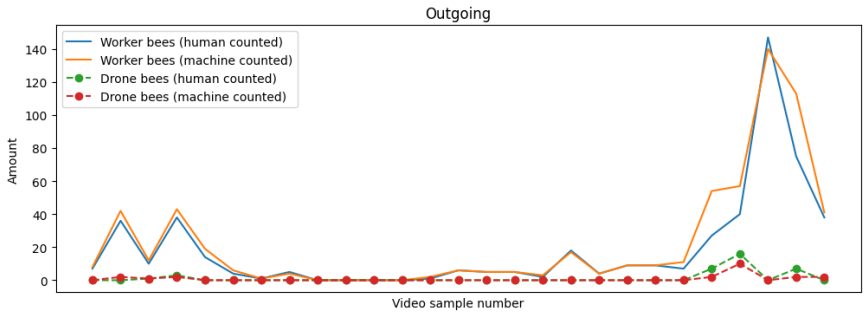
To manually count the bees, we used a hand tally counter to keep track of the numbers. We watched the bee videos at half the normal speed to facilitate the identification of bees entering and leaving the hive. The counting process involved focusing on one class at a time (e.g., drones, workers, and potential enemies). Before commencing the counting, we established predefined rules to determine when a bee should be considered as entering or leaving the hive. These rules align with those described earlier in Subsection 2.7. Bees that were guarding the hive or simply walking outside were not included in our count, as they were not relevant to our study.

Furthermore, we needed to devise a method for calculating the error in our measurements. Since our data sometimes included a true count of zero, using a simple formula such as $|real - measured| \cdot real^{-1}$ would result in an infinite value if the real count is zero. To overcome this issue, we utilized a metric known as MAAP (Mean Arctangent Absolute Percentage) [Hyndman and Koehler 2006]. The MAAP metric addresses this problem by approaching zero instead of infinity, enabling more accurate error calculations even when the true count is zero. The error calculation is performed separately for each combination of class and flow (e.g., incoming workers, and outgoing drones).

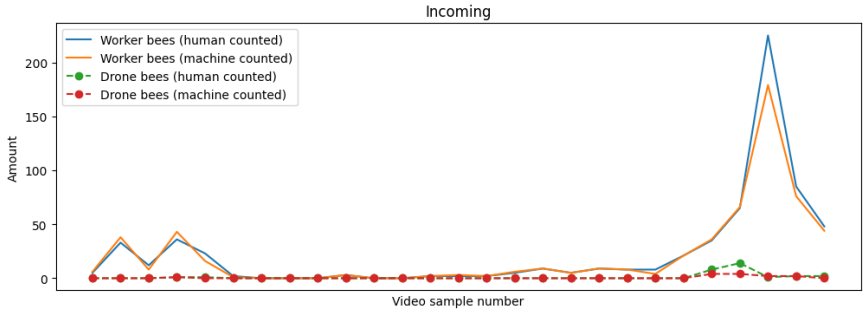
3. Results and Discussion

3.1. Prediction precision

We looked at 27 videos of bees entering and leaving hives and counted the number of drones and workers bees in each video. We compared our manual counts (real) with the counts generated by ApisFlow (measured). However, because there were very few instances of enemies invading the hive, we could not test the algorithm's accuracy in counting them; we can only prove the efficiency of counting drones and workers.



(a)



(b)

Figure 7. Comparison between real and measured data (a) human counted *versus* machine counted outgoing flow. (b) human counted *versus* machine counted incoming flow.

Table 2. Class errors

	Worker	Drone
Incoming	3.7%	4.0%
Outgoing	6.1%	5.9%

Table 2 and Figure 7 show a minor error across all classes compared to recent video bee counters as discussed by Odemer (2022). Please note a difference of approximately 2% between the incoming and outgoing bees in each class. This difference may be attributed to the increased challenge of predicting when a bee will start flying out of the beehive, compared to when it enters. Outgoing bees, being stationary at the entrance and departing randomly, pose a greater difficulty for prediction. Nevertheless, the impressive results hold significance, particularly due to our accomplishment of real-time performance and low-resource applications.

3.2. Prediction time

Since we are working with videos with a frame rate of 30 frames per second (fps), the ApisFlow tool must also run at a speed of at least 30 fps. The algorithm's performance will depend directly on the tools used (Section 2.2).

ApisFlow perfomed at 38.9 fps average on our machine, with a 12.9 fps standard deviation. Based on that, we can see that, on our machine, the ApisFlow ran, on average, faster than the data supplied, meaning that it can be considered a real-time algorithm.

To achieve real-time processing, we made several improvements to the tracking algorithm. These included using faster array access in NumPy, minimizing the use of vector and matrix operations, and identifying which parts of the program took the most time, trying to speed them up. As a result, we could process our part of the algorithm in an average of 2 ms. However, the object detector took up most of the processing time, which took an average of 13 ms. Since ApisFlow is not tied to a specific object detection algorithm, as new research on object detection progresses, we can easily incorporate faster algorithms and achieve speedier processing rates in frames per second.

4. Conclusion

This paper's main contribution is the improved recognition of worker bees and drones, making bee counting, tracking, and classification easier, safer, faster, and more precise for beekeepers. The ApisFlow framework enables real-time remote access, opening doors for advancements in Precision Beekeeping. It enhances our understanding of honey bee flow within beehives, providing valuable insights into predicting colony swarming, assessing colony health, identifying potential threats, evaluating pollination potential, detecting laying workers, and recognizing deficits in returning foragers.

As a suggestion for future work, we propose integrating a particle filter to improve the tracking of bee movement and overall accuracy. Furthermore, to enhance drone detection, we recommend exploring the combination of object detector classification with ratio proportion classification [O'Brien et al. 2022].

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Danielo G. Gomes (processes 311845/2022-3 and 432585/2016-8), Gabriel Fruet (process 144754/2022-3), and Rafael Capelo (process 162249/2022-5) thank the financial support of the Conselho Nacional de Desenvolvimento Científico e Tecnológico-Brasil (CNPq). Antonio Rafael Braga thanks for the financial support of the Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico - FUNCAP, process BP5-00197-00220.02.00/22. The authors would like to express their gratitude to Dr. Rahman Tashakkori for providing the video dataset used in this paper.

References

Albuquerque, D. Q., Braga, A. R., Bomfim, I. G. A., and Gomes, D. G. (2022). Aplicando um modelo yolo para detectar e diferenciar por imagem castas de abelhas melíferas de forma automatizada. In *Anais do XIII Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 51–60. SBC.

Andrijević, N., Urošević, V., Arsić, B., Herceg, D., and Savić, B. (2022). Iot monitoring and prediction modeling of honeybee activity with alarm. *Electronics*, 11(5):783.

Barros, C., Freitas, E. D., Braga, A. R., Bomfim, I. G., and Gomes, D. (2021). Aplicando redes neurais convolucionais em imagens para reconhecimento automatizado de abelhas melíferas (*Apis mellifera* l.). In *Anais do XII Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 19–28, Porto Alegre, RS, Brasil. SBC. https://doi.org/10.5753/wcama.2021.15733.

Boes, K. (2010). Honeybee colony drone production and maintenance in accordance with environmental factors: an interplay of queen and worker decisions. *Insectes sociaux*, 57:1–9.

Delaplane, K. S., Dag, A., Danka, R. G., Freitas, B. M., Garibaldi, L. A., Goodwin, R. M., and Hormaza, J. I. (2013). Standard methods for pollination research with apis mellifera. *Journal of Apicultural Research*, 52(4):1–28.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

Hadjur, H., Ammar, D., and Lefèvre, L. (2022). Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services. *Computers and Electronics in Agriculture*, 192:106604. https://doi.org/10.1016/j.compag.2021.106604.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688.

Khalifa, S. A., Elshafiey, E. H., Shetaia, A. A., El-Wahed, A. A. A., Algethami, A. F., Musharraf, S. G., AlAjmi, M. F., Zhao, C., Masry, S. H., Abdel-Daim, M. M., et al. (2021). Overview of bee pollination and its economic value for crop production. *Insects*, 12(8):688.

Odemer, R. (2022). Approaches, challenges and recent advances in automated bee counting devices: A review. *Annals of Applied Biology*, 180(1):73–89.

O'Brien, W., Tashakkori, R., Parry, R. M., Hamza, A., and Graber, J. (2022). Estimating the number of drones at the entrance of a honey bee hive using machine learning tools. In *SoutheastCon 2022*, pages 397–404.

Sagili, R. R., Burgett, D., et al. (2011). Evaluating honey bee colonies for pollination: a guide for commercial growers and beekeepers.

Yang, F., Zhang, X., and Liu, B. (2022). Video object tracking based on yolov7 and deepsort. *arXiv preprint arXiv:2207.12202*.

Yin, Y., Li, H., and Fu, W. (2020). Faster-yolo: An accurate and faster object detection method. *Digital Signal Processing*, 102:102756.