Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

FORMIGOMETRO: YOLO BASED VIDEO TRACKING TUTORIAL

Goal of this tutorial: help anyone develop their own object detection model using YOLO for their research or troubleshoot issues regarding video-tracking or video behavioral analysis. Studying leaf-cutter ant's foraging behavior, the available tools did not prove to be sufficient for addressing our research question, pointing us to the need to develop our own tool with sufficient flexibility for future use.

MACHINE LEARNING TECHNIQUES APPLIED TO BEHAVIORAL RESEARCH (ADAPTED FROM MY MASTER'S DISSERTATION)

Machine learning is defined as the study of algorithms that allow computers to perform tasks automatically and improve their performance thanks to previous data. Machine learning uses neural networks and is the basis of artificial intelligence (Naqa and Murphy, 2015; Zhang, 2017). The modern and globalized world uses machine learning techniques and artificial intelligence (AI) to perform several activities, from social media on our cellphones to defining public policies based on big data. Whether we understand it or are aware of this presence, it is everywhere.

Machine learning uses input data to extract features that can "learn" and perform a task based on this learning. Machine learning can be divided into two main groups: shallow and deep learning (Figure 1). The main difference between shallow and deep learning refers to feature extraction. While deep learning involves a more complex architecture and an automatic selection of features, in shallow learning, humans select the features that will be used for training the models. In this sense, deep learning offers an important tool to solve and deal with many biological questions since variability is an intrinsic component of biological data. Selecting the best features for model training may be challenging when researchers do not have enough knowledge about their system of study.

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

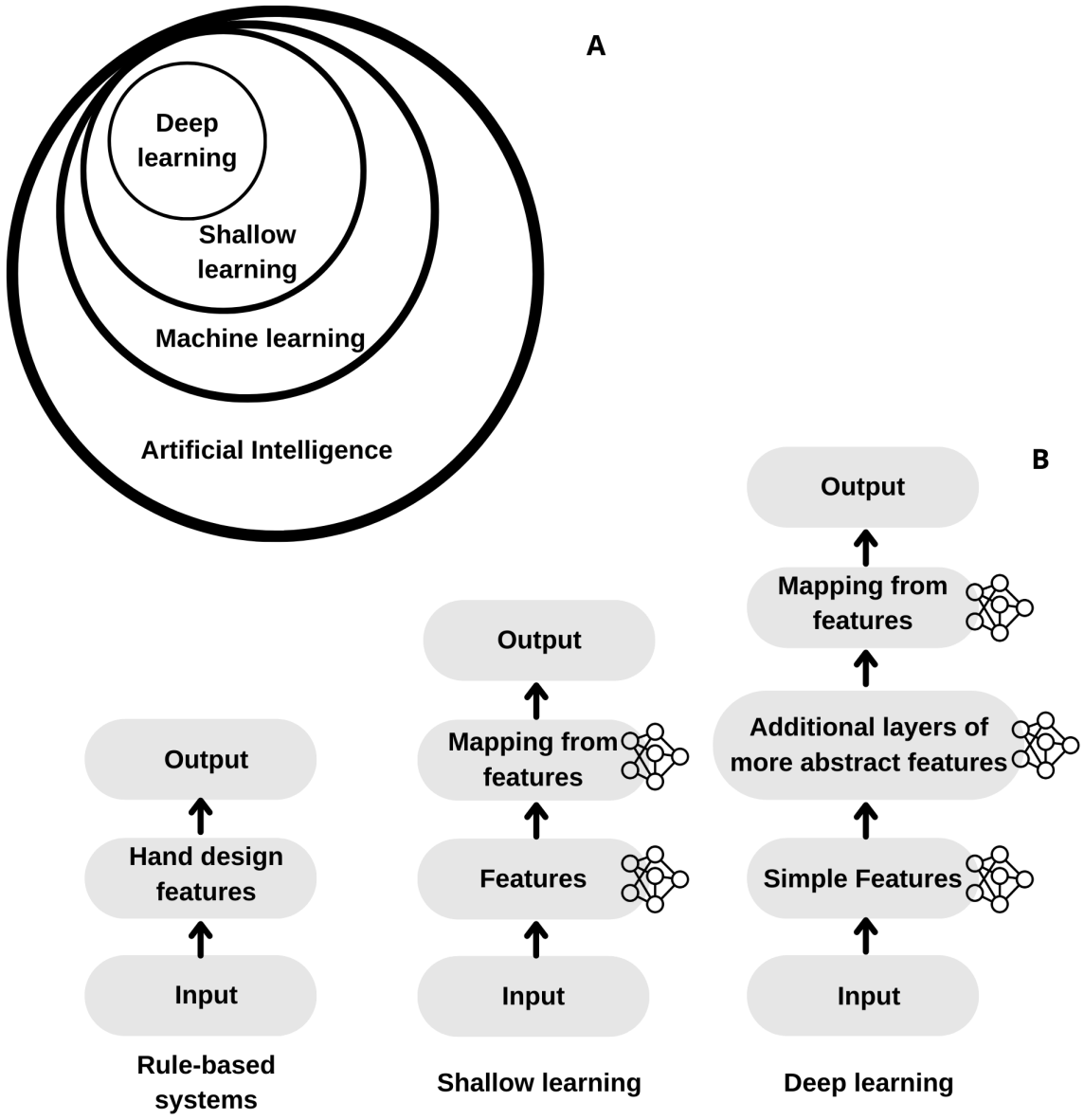


Figure 1) Schematic representation showing machine learning structure. Venn diagram of the relationship between different components of Artificial Intelligence. Flow chart of different AI parts used in machine learning. Small network symbols indicate components that can learn from data (adapted from Goodfellow et al., 2016).

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

OBJECTIVES

1. Developing a video-tracking software able to analyze the foraging behavior of a leaf-cutter ant colony inside the laboratory.
2. Applying this video-tracking tool to a synchronization experiment using a leaf-cutter ant colony. We also aim to monitor the daily leaf consumption and leaf-cutting rhythm in the same experiment.

Objective 1 is treated briefly below (for more detailed information, please check the Tutorial present in the Annexes section). Objective 2 is the central theme discussed in Chapter 2 of this manuscript.

DEVELOPMENT OF AN OBJECT DETECTION AND COUNTING TOOL FOR ANALYZING ANT'S FORAGING ACTIVITY

Our group has been developing tools for analyzing ant foraging trails through video-tracking, allowing size discrimination, and monitoring a population of about 10-15 thousand individuals. This software uses machine learning and deep learning techniques to automate the counting of high flows of ants on foraging trails, being an essential tool for studies with this type of system, enabling deeper investigation of temporal aspects at the colony level (Toledo, 2014 and 2018). For developing our final video-tracking tool, many steps were involved in the machine learning process (Figure 2).

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

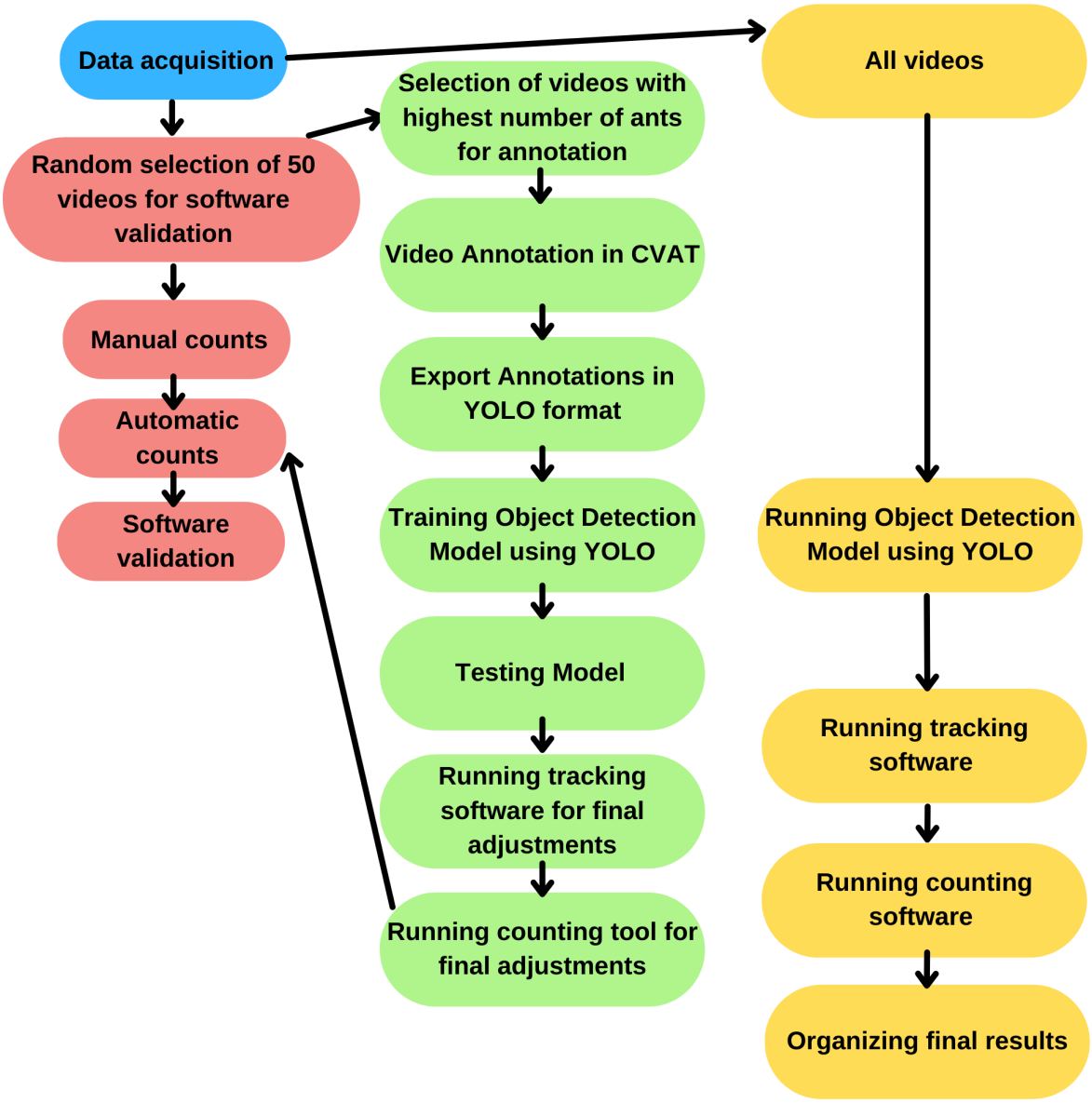
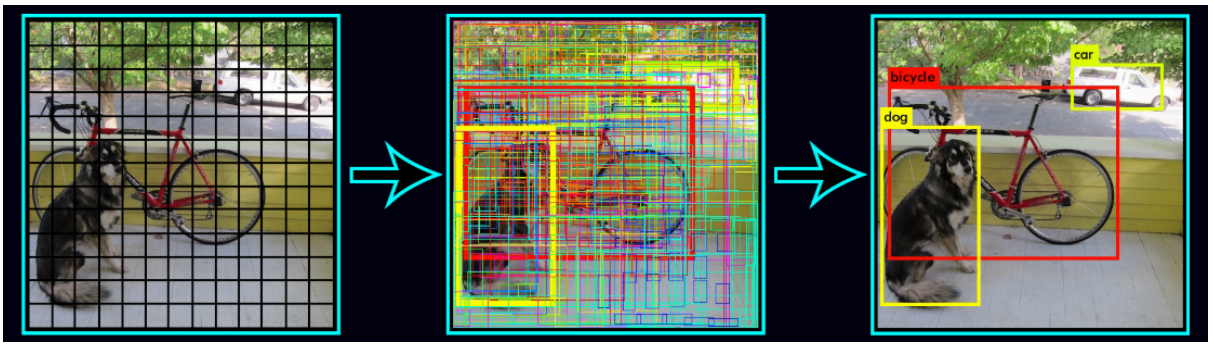


Figure 2) Flowchart of all the steps involved in our video-tracking software development. Blue box involves data acquisition (experiments). Red boxes show all steps involved in the software validation stage. Green and yellow boxes show the deep learning train model stage and run our model results for all our data.

The main challenges regarding using this type of tool involve object detection. In this case, it is necessary to make the computer recognize the object (ant) and follow it throughout the video, accounting for it in the flow counts. However, the foraging trail is a complex environment that involves different elements, albeit in the laboratory context. We have, for example, ants of various sizes and functions, such as those seeking or carrying leaves, as well as those removing obstacles ("waste")

from the trail itself. In addition, the flow of ants can be very high at certain times, especially when the colony is feeding and during the night. Thus, this software needs to be trained to recognize the ants in different frame contexts in the video image. It is precisely in developing this kind of tool that part of the present work took place, making use of powerful object detection techniques such as YOLOv2 (You Only Look Once - versão 2, Redmon *et al.*, 2016) and YOLOv5 (You Only Look Once - version 5, Zhu *et al*., 2021). When detecting objects in a video, this technique observes each frame once and calculates the probability that each object found in the frame belongs to a previously trained class (Figure 3). This way allows the analysis of a large amount of video data, being an important tool used, for example, for monitoring public security cameras.



https://towardsdatascience.com/implementing-yolo-on-a-custom-dataset-20101473ce53

Figure 3: Frame-by-frame object detection mechanism realized by YOLO.

During this project, we developed two versions of this software, and they required several complex steps which are detailed and organized in our supplementary material. The first version used YOLOv2 and counted ants in our pilot experiment with *Acromyrmex* ants (Figure 4). However, in our main experiment using *Atta* ants, the number of ants in the trail became extraordinarily high, reaching hundreds of ants per 2 minutes video. Since our first version was powerful enough to detect and count until 50 ants per video, and the software error increased the higher the number of ants per video, we decided to develop a second version of the software using one of the most powerful and recent object detection techniques ever developed, which was YOLOv5 (Zhu *et al*., 2021; Fang *et al.,* 2021).

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

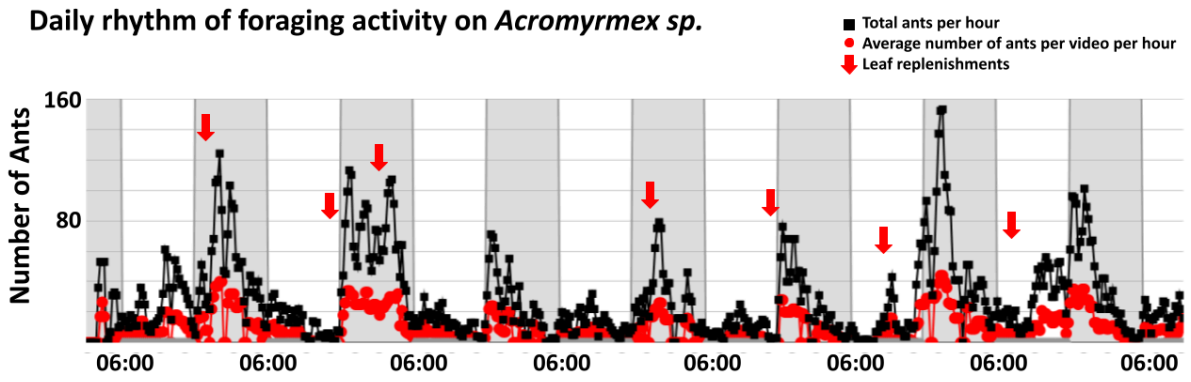


Figure 4: Daily rhythm of foraging. Recordings of 8 days of activity of a colony of Acromyrmex sp. exposed to a 12:12 LD cycle (lights on at 06:00h). Humidity was kept at 60% and room temperature at 23ºC. Leaf replenishments occurred randomly once a day.

YOLOv5 provided us with great results, as we were able to count over 143.000 ants during the main experiment. Nevertheless, it is essential to acknowledge that developing a second version of the software in such a shorter time (we developed version 1 during a year and version 2 in 4 months) and obtaining these results was only possible due to a recently open-source project for data annotation, called Computer Vision Annotation Tool (CVAT - Intel corporation). CVAT enables video annotation using a tracking mode function and online annotation that can be accessed from different devices simultaneously. Our software was developed using data from our video experiments. A precise object annotation in each image plays a vital role in software performance. Poor data annotation makes machine learning models decrease their accuracy, and it may be computationally hard to boost their performance with optimization tools.

We emphasize the importance of data selection for the annotation step, as it is important for biologists to select data that represents our object of study satisfactorily. In our first version of the software, we faced two issues. First, we transformed all our data in binary and ran algorithms that extracted 10.000 frames (the minimum required to train an object detection model) according to the probability of containing objects that could be identified as ants. Our pilot experiment did not contain a high number of ants per video (only 16% of the videos contained more than 20 ants), but also many video frames did not contain any objects (27% of the videos had 0 ants). Thus, a significant number of frames extracted by the algorithm

had no ants. That was a second issue because a lot of the images used to train the YOLOv2 model could not provide a higher pool of ant annotations, making the model accuracy smaller. When we developed the second version of the software using YOLOv5 and CVAT, we selected a pool of videos with a high number of ants from both light and dark phases. Since we knew how computationally powerful YOLOv5 was, we trusted it would solve videos with smaller numbers of ants and trained our model on the worst-case scenario: hundreds of ants per video. We chose to share this information here because we believe that many ethological researchers using video-tracking software may not understand the importance of clear image data (with high contrast between image background and objects and no unnecessary objects in the video), as machines are not able to solve issues that are easily detected by the expert human eye. Also, we believe sharing this experience may help programmers who collaborate with biologists to develop tools to solve challenging experimental questions such as ours. In our experience, YOLOv2 allowed us to count 5052 *Acromyrmex* ants in our pilot experiment (1152 minutes of video), while YOLOv5 allowed us to count 143.904 *Atta* ants in our final experiment (4320 minutes of video recordings).

```
SYSTEM REQUIREMENTS:

This tutorial was elaborated for Ubuntu 20.04 LTS.

HARDWARE REQUIREMENTS:
GPU (NVIDIA GTX 1060 in our case)
16GB RAM (we used 32GB)

SOFTWARE REQUIREMENTS:

All codes and Readme.md files are available on GitHub
labcog/Contador repository.

### BASIC USAGE (= local copy)###
git tutorial: http://rogerdudler.github.io/git-guide/

1. Download the current version:
* if you are downloading it for the first time, you must
clone the repository:

#!git
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

```
$ git clone
https://bitbucket.org/toddy757/labcog/branch/contador
PS: PRECISA AJUSTAR COMO VAMOS DISPONIBILIZAR O REPOSITÓRIO
(Não me recordo se ele fica fechado para edição para todo
mundo que não faz parte do repo do labcog. E que eu me lembre
precisavamos mover ele pra master ao invés de branch, certo?


#!git
$ git pull origin master
 All the alterations that you do will only be saved locally.


2. If you want to delete your modifications and download the
current version again:
#!git
$ git fetch --all
$ git reset --hard origin/master


# yolo
v5: requirements.txt from repositor ultralytics/yolov5
v3: opencv (gpu e módulo dnn linkado
pandas

## dataRRANSAC
opencv-python
matplotlib
pyaml
numpy

# dataCountline
numpy
pyaml

# optimization (we do not talk about the optimizers in this
tutorial, but some information is available on the repository
contador)
bayesian-optimization
motmetrics
scipy
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

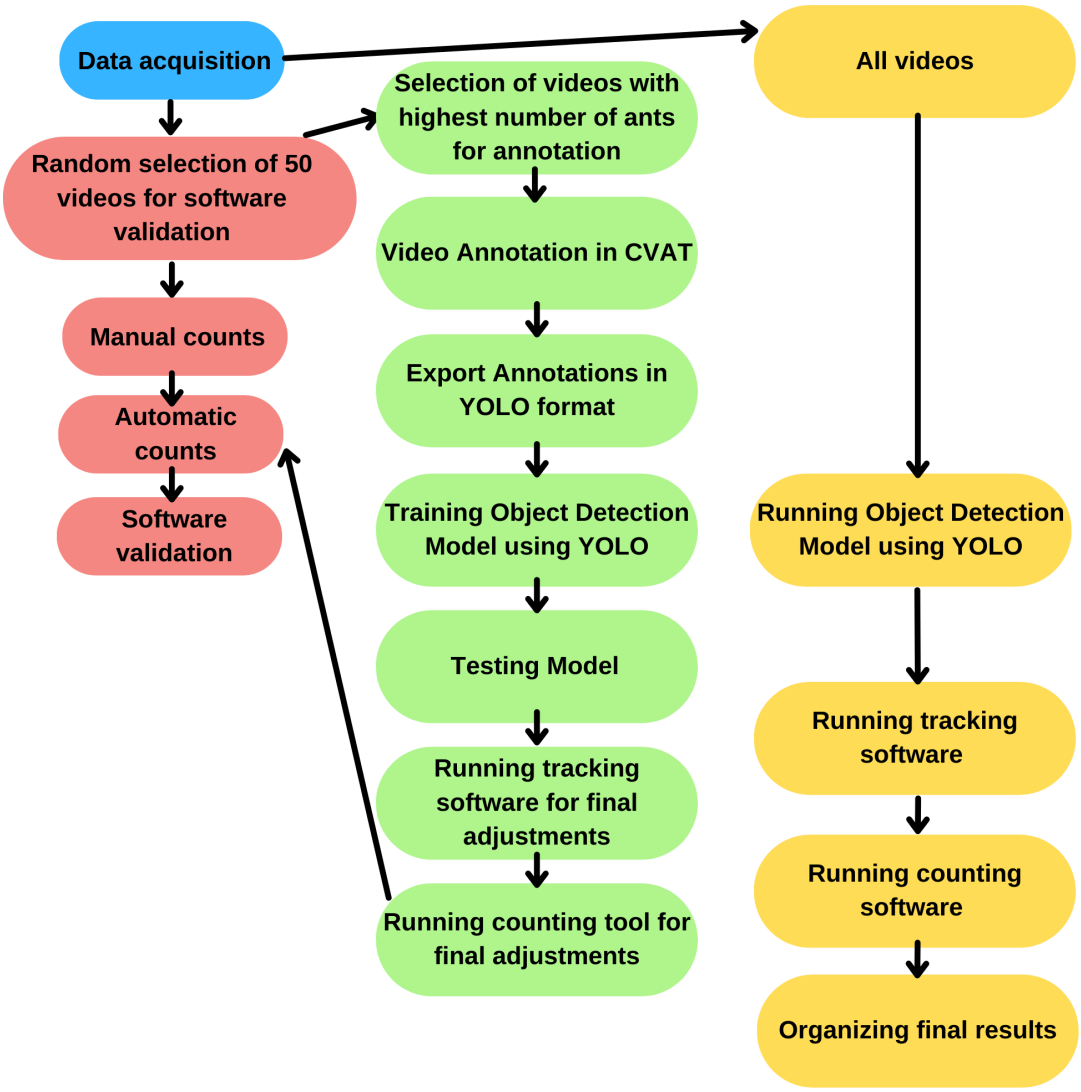


Figure 1: Flowchart of all the steps involved in our video-tracking software development. Blue box involves data acquisition (experiments). Red boxes show all steps involved in the software validation stage. Green and yellow boxes show respectively the deep learning train model stage and running our model.

1. **Data acquisition**
2. **Software validation**
3. **Data annotation**
4. **Training model**
5. **Testing the object detection model**
6. **Tracking Tool**
7. **Counting Tool**
8. **Analyzing your data:**
    a. **Organizing the data**

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

2. **Running Object Detection Model for all videos**
3. **Running Tracking tool for all videos**
4. **Running Counting tool for all videos**
5. **Some tips and reading suggestions**

1. Data Acquisition:

If you plan on using video-tracking tools to analyze your behavioral data, you must record your data in a way that maximizes the probability of the software's success. That means that video quality, recording angle, recording distance, and focus are all essential features. They should not change across your video data. We strongly recommend recording your video data from an angle that does not include unnecessary objects in the images (for example, shadows, reflections, and other objects that will not be analyzed).

Video tracking tools usually contrast the background and the object of interest to identify the object and track it. For behavioral research, that means: if your animal model has light colors, you should use darker backgrounds; if it has dark colors, you should use lighter backgrounds. Also, it would help if you thought about recording time: the longer your videos are, the longer is the time required to analyze them. Also, longer videos usually occupy a lot of memory storage. If you need to compress the video data to save it, you will lose image quality, which can be a problem. We also recommend checking the video formats available to your recordings (we prefer .mp4 or .mkv). Also, the more objects you need to track, the harder it is to obtain a high accuracy from most video-tracking tools. All this information and tips are important because video-tracking programs perform poorly when analyzing video recordings containing noisy images.

Using iSpy for scheduling recordings:

- Download iSpy: https://www.ispyconnect.com/download.aspx (it runs only on Windows. Agent DVR is another tool from the same group that runs on other operating systems, but we have not tested it. CHECK IF YOUR VERSION SHOULD BE 32bits or 64bits)
- Install it.

- Run iSpy and add all your recording devices. By right-clicking, you should set all the recording features to that camera and schedule your recordings. Pay attention to which directory your data is being saved.
- iSpy user guide: https://www.ispyconnect.com/userguide.aspx

2. Software Validation:

- From all your video data, randomly select a group of videos for software validation (20-50) videos. If you have different recording conditions (in our case, we had videos recorded under light and darkness conditions), make sure that your selected recordings represent all your possible conditions.
- In our case, we manually counted the number of ants in each of these selected videos using the software **contadorManual.py**

contadorManual.py

This program opens the video with a blue line drawn in the middle of the screen and allows manual counting as follows:
The count takes place when the ant crosses the line.
If it crosses the line in the top direction of the video: UP
If it crosses the line in the bottom direction of the videos: DOWN
or
If it crosses the line in the right direction of the video: RIGHT
If it crosses the line in the left direction of the videos: LEFT

To run it (at the directory that contains it): python contadorManual.py
Parameters:
-video (required)
[- h]: optional (help)
- line [LINE LINE LINE]: allows you to set 4 parameters to adjust the position of the line (x,y, and x,y coordinates)

COMMANDS:
SHIFT: Play/pause
W: Count Up
S: Count Down

+: accelerate video
-: Decreases the video speed
(you can also set different keys for each command in the software code).

It will save a .txt file with the results.

EXAMPLE:
```
$ python contadorManual.py -video
/PATH_TO_VIDEO/VIDEONAME.mp4
```

- Run contadorManual.py for all your select videos for validation (we recommend saving them in a single directory).
- Organize all .txt result files into a single file (.csv)
- Here we stop the validation process and proceed with the Data Annotation. We will return to it after training and testing the model.

3. Data Annotation:

- Select the videos manually counted with the highest number of ants (10.000 video frames are the minimum required. It is crucial to select videos from all your experiment conditions. In our case, we selected four videos, two from the dark phase and two from the light phase).
- We ran all our data annotations in https://cvat.org/.
- After creating our profile, we created a project, defined the object class for annotation (ants and ants+leaf), and created a task for each video. You can organize the data in other ways.

NOTE: Data annotation is probably the most exhaustive part of this tutorial, but it is also one of the most important. We recommend CVAT because it allows you to have help during annotations and stop and continue your annotations. We took almost a month annotating the data working about 8-10 hours per day, so plan your time accordingly.

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

CVAT TUTORIAL

CVAT (Computer Vision Annotation Tool) is a tool developed by Intel co. and is open source. In it, you can upload images for annotation and do the annotation directly on video, using one or more classes and with various annotation features. It allows you to download these image annotations in different formats, depending on which one you will use to train your network. CVAT also allows you to start and stop a video annotation at any time.

CVAT can be run in two ways: via the website (cvat.org), and the other is installation via GitHub repository, running through docker with port forwarding. The difference is that the one via the website has an upload limit of 500mb per user.

Note: CVAT via GitHub has access to openVino, an automatic annotation tool, and its docker images occupy a lot of storage(you will need at least 15GB of free disk space).

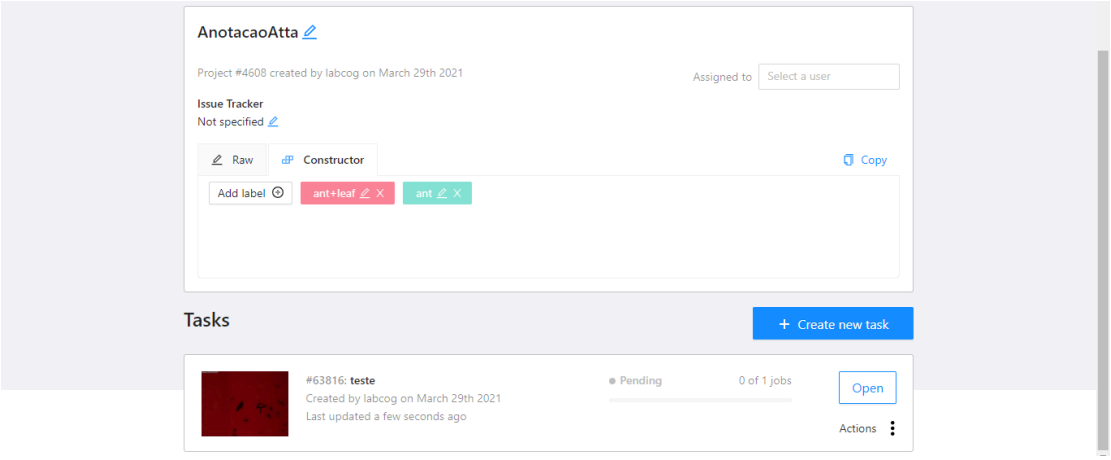
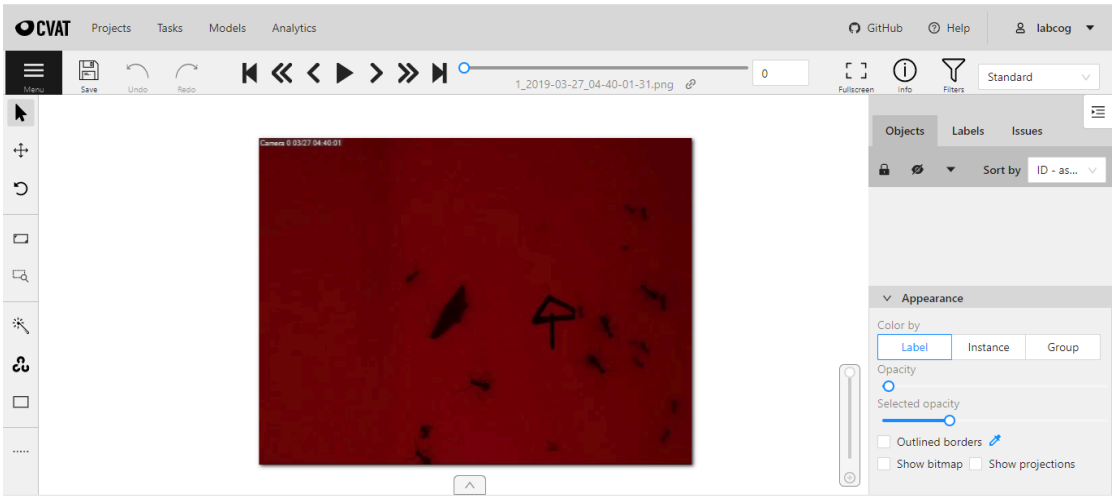


Figure 2: CVAT online.

How to annotate images on CVAT:

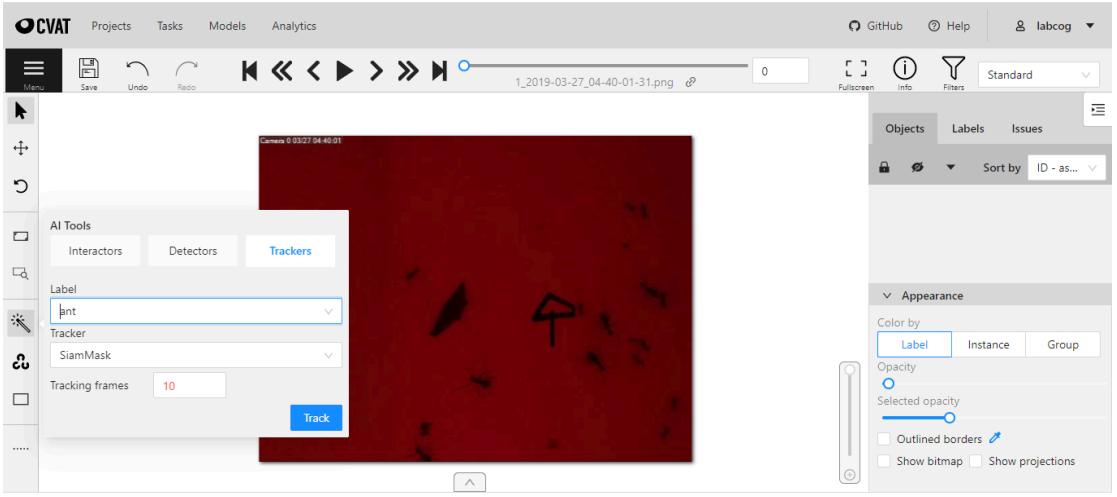
- Click **Open the task** and then click **Job #NUMBER**
- The annotation tool should open with the images associated with that task.

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024



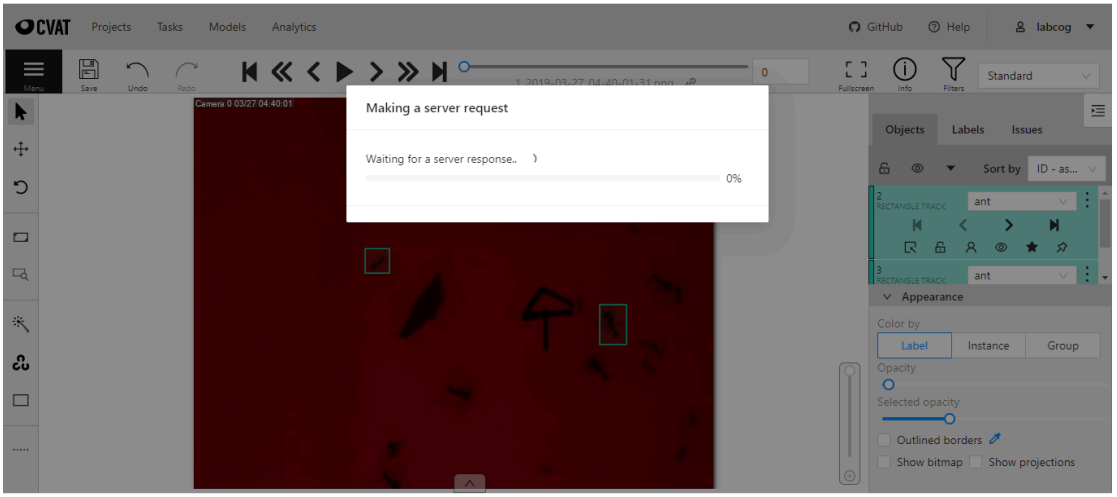
- There are many ways to annotate the images, and the options are in the left corner of the screen. In the right-hand corner, you will see the objects you have annotated in the "Objects" section as you do so. Here we have the automatic annotation recommendation.
    - Using annotation with the tracker function.

This function is intended to annotate objects every n frames automatically. To use it, you will annotate the object using the cross-like manual annotation and wait while the system cycles through the following frames and annotates the position of that object as it changes in the video. This process may take a while. To use this function, click on the "**magic wand**" button and select the "**tracker function**":
Select the class you want to annotate, change 10 to 4 frames (it will be faster to process and probably contain fewer errors), and click "**track**".



Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

To make an annotation, using a cross on the tip of the mouse, you will click on the image and select the area around the ant. IMPORTANT: SELECT THE WHOLE ANT WITHOUT LEAVING TOO MUCH SPACE OUTSIDE.

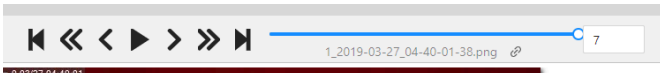


You will see that it has automatically annotated that ant in the subsequent four frames when you finish. Annotate all the ants in that frame.

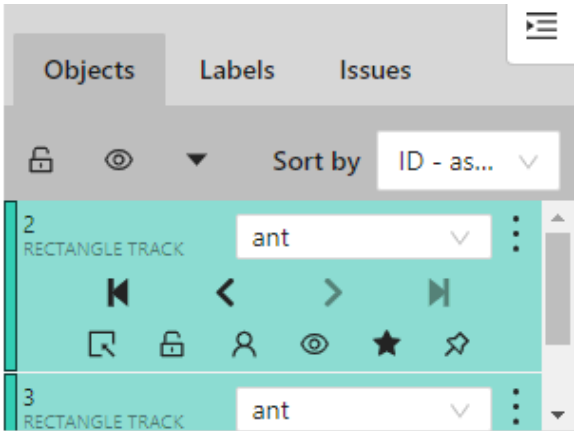
Go through the following image frames correcting the annotation if there are any errors (probably depending on the ant's position, the rectangle may be of inadequate size, so it is worth adjusting or moving it if the ant has moved). Each ant gets an id number (right side of the screen).
To make a new annotation, press the "n" key on your keyboard, and you will see the marking cross on the tip of your mouse again.

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

These little arrows allow you to navigate between the frames of the video.



In the first frame that an ant enters, you should start the annotation, even if it is just partially in the image.
As soon as the ant disappears from the image, you should click on the box (see the object's id), go to the Objects section to find the object with that id, and click on the square with a small arrow ("Switch Outside Property"). This function indicates to the image that that object has left the video. If you delete the box, it will understand that that object NEVER EXISTED and delete all annotations for that object.



Note: Remember that the id number says that the object is unique (same ant). If the same ant is not accurately identified (more than one id number for the same object or the same id number corresponding to a different object), the tracking will not work.
Note: Ants interact a lot. Use the zoom mode to make sure where your annotations are accurate.
Note: Save often. I usually save after using the tracker for each new ant and every ten frames by correcting/adjusting the boxes (there is also an option on CVAT that saves your annotations automatically, but I also recommend manually saving it to be safe).

4. Training Model:

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

- We used YOLOv5 (a state-of-the-art real-time object detector system) to train the object detection model.
- Download all the annotations from CVAT in the format YOLO to train the YOLOv5 network.
- You will also need to download all the annotations in the CVAT video (.xml) format for adjustments in the tracking mode (RRANSAC - will be treated later).
- All the following steps will require GPU to run. Our graphic card model was NVIDIA GTX 1060
- Install the NVIDIA driver **(some of these steps might change according to your NVIDIA model)**

```
$ wget -c
https://us.download.nvidia.com/XFree86/Linux-x86_64/450.80.02
/NVIDIA-Linux-x86_64-450.80.02.run

$ wget -c
https://developer.nvidia.com/compute/cuda/10.0/Prod/local_ins
tallers/cuda_10.0.130_410.48_linux
   1) $ chmod +x  for the first file
   2) $ ls Downloads
   3) Run: sudo ./NVIDIA..(name) --dkms
   4) Follow the instructions on the screen
         a) If an error occurs:
            i)   $ more /etc/X11/default-display-manager (it
                 returned -gdm3, but it could other display
                 manager)
                 $ sudo systemctl stop gdm3
                 $ sudo ./NVIDIA-Linux-x86_64-450.80.02.run
                 --uninstall
         b) Reboot
         c) Repeat the installation process
   5) Check GPU status
$ nvidia-smi
It should return something like this if everything is
working.

mila@mila-System-Product-Name:~$ nvidia-smi
Wed Jan 15 11:55:05 2020
+-----------------------------------------------------------
----------------+
| NVIDIA-SMI 435.21    Driver Version: 435.21      CUDA
Version: 10.1    |
|-------------------------------+----------------------+------
----------------+
| GPU  Name      Persistence-M| Bus-Id       Disp.A | Volatile
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

```
Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util
Compute M. |
|===============================+======================+======
================|
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0 Off |
      N/A |
|  0%   47CP5    7W / 120W |169MiB /  3019MiB |    2%
Default |
+-------------------------------+----------------------+------
----------------+


+-----------------------------------------------------------------
----------------+
| Processes:
      GPU Memory |
|  GPU     PID   Type   Process name
Usage      |
|=================================================================
================|
|     0     2223       G   /usr/lib/xorg/Xorg
      96MiB |
|     0     2411       G   /usr/bin/gnome-shell
      70MiB |
+-----------------------------------------------------------------
----------------+
```

- You might also need to install the Cuda toolkit:

Installation (item 3.8 Ubuntu):
https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html#ubuntu-x86_64

```
$ sudo sh cuda_<version>_linux.run
```

1. Create a file at /etc/modprobe.d/blacklist-nouveau.conf with the following contents:

   ```
   blacklist-nouveau
   options nouveau modset=0
   ```

2. Regenarate the kernel initramfs:

   ```
   $ sudo update-initramfs -u
   ```

3. Reboot

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

- YOLOv5 Repository (please check the Readme.md file for instructions): https://github.com/ultralytics/yolov5
- 10000 training cycles are sufficient for small data sets, but the ideal, especially for larger data sets, is 100000 cycles.
- You can find all the scripts described here in: https://drive.google.com/drive/folders/1PBiS48w303iVhtvE3M0aBI7ZgWXO6c9s?usp=sharing
- Run the exp.sh
    - It is a bash script that generates all the other files

  In this case, it takes images in 4 folders (each one corresponds to an annotated video), mixes the images, and separates them between training and validation so that all videos are represented in both the training and the validation set. In the end, it generates a list of images and labels with this training set and another one with the test set.
  Output of exp.sh: shows for each folder (video) how many images are used in training and in validation

```
##exp.sh##

#!/bin/bash
DATAPATH="/home/marcelo.arruda/Projetos/M/datasets/videosComplicados/"
#DATAPATH=""
V1="1_2019-03-27_04-40-01"
V2="1_2019-04-04_23-00-00"
V3="1_2019-04-05_06-40-00"
V4="1_2019-04-08_12-00-01"

echo "
# ReTreinar Yolo em VideosComplicados
"

echo "
## Dados
    1_2019-03-27_04-40-01 (treino)
    1_2019-04-04_23-00-00 (validação e teste)
    1_2019-04-05_06-40-00 (validação e teste)
    1_2019-04-08_12-00-01 (treino)
"

printVideoDSInfo () {
    video=$1
    datapath=$2
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

```
    echo "- ${video}:
    - images: $(ls "${datapath}${video}"/*.png | wc -l)
        $(ls "${datapath}${video}"/*.png | head -1)
    - txts:   $(ls
"${datapath}${video}"/[0-9]*[0-9].txt | wc -l)
        $(ls "${datapath}${video}"/[0-9]*[0-9].txt |
head -1)
    - list:   $(cat
"${datapath}${video}/list_${video}.txt" | wc -l) lines
        $(cat "${datapath}${video}/list_${video}.txt" |
head -1)
        $(cat "${datapath}${video}/list_${video}.txt" |
tail -1)
    "
}

printVideoDSInfo $V1 $DATAPATH
printVideoDSInfo $V2 $DATAPATH
printVideoDSInfo $V3 $DATAPATH
printVideoDSInfo $V4 $DATAPATH


echo "
## Treino:
    ${V1}: $(cat "${DATAPATH}${V1}/list_${V1}.txt" | wc
-l) +
    ${V4}: $(cat "${DATAPATH}${V4}/list_${V4}.txt" | wc
-l) "
cat "${DATAPATH}${V1}/list_${V1}.txt"
"${DATAPATH}${V4}/list_${V4}.txt" > list_train.txt
echo "    -> list_train.txt $(cat list_train.txt | wc
-l)"


echo "
## Validação e teste:
    ${V2}: $(cat "$DATAPATH""${V2}/list_${V2}.txt" | wc
-l) +
    ${V3}: $(cat "$DATAPATH""${V3}/list_${V3}.txt" | wc
-l) "

cat "${DATAPATH}${V2}/list_${V2}.txt"
"${DATAPATH}${V3}/list_${V3}.txt" > list_testval.txt
shuf list_testval.txt -o list_testval.txt
head -n 1000 list_testval.txt > list_validation.txt
tail -n +1001 list_testval.txt > list_test.txt
echo "    -> list_validation.txt $(cat list_validation
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

- Run the fullTrainingVComplicados.sh
    - YOLO training script on the lists listTrain.txt and listVal.txt. The execution of this script will actually produce a trained model.

```
#fullTrainingVComplicados.sh#

rush=/usr/local/bin/rush

find "$1" -name "*[0-9].txt" | $rush "python
multilabelsYoloAnnotation.py {} {}"
```

- fullTrainingVComplicados.yaml: contains all especifications from the YOLO model

5. Testing the object detection model

- Install docker: https://docs.docker.com/v17.09/engine/installation/linux/docker-ce/ubuntu/
- Building the dockerfile and open a new image: (tutorial https://www.youtube.com/watch?v=LQjaJINkQ)

```
$ make cudnn_tensorflow_opencv-10.2_2.2.0_4.3.0
$docker images

output example:

REPOSITORY                                TAG                                 IMAGE ID            CREATED             SIZE
datamachines/cudnn_tensorflow_opencv      10.2_2.2.0_4.3.0-20200615           d58af4a715a4        14 hours ago        5.86GB
nvidia/cuda                               10.2-cudnn7-devel-ubuntu18.04       2b8bb5f68029        6 days ago          3.82GB
$ xhost +local:docker
$ docker run -ti --rm --gpus all -e DISPLAY=$DISPLAY -v
/tmp/.X11-unix/:/tmp/.X11-unix --ipc host --device /dev/nvidia0:/dev/nvidia0
--device /dev/nvidiactl:/dev/nvidiactl --device /dev/nvidia-uvm:/dev/nvidia-uvm
-v /home/:/home/ d58af4a715a4 /bin/bash (the image id varies according to your
computer)


if an error occurs:
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

"docker: Error response from daemon: could not select device driver "" with capabilities: [[gpu]]."

```
$ sudo apt install nvidia-container-runtime
```

PS: you might need to install some packages inside the docker container:

```
$ apt-get update
$ apt-get upgrade
$ apt install python3
$ apt install python3-pip`
$ python3 -m pip install --upgrade pip
$ python3 -m pip install numpy`
$ python3 -m pip install opencv-python
$ python3 -m pip install opencv-python==3.2.0.8 (this version may change)
$ python3 -m pip install pyaml
```

PS: --GPU is the option that makes the computer run using the GPU instead of the CPU. You can check the GPU and CPU usage while yolov5Detect.py is running to verify "how much it is taking from the computer." If you do not use the --GPU parameter, yolov5Detect.py will run using only the CPU, which will be slower and require a lot of computer power.

to check CPU usage on Ubuntu:

`$ htop` (shows CPU usage, if the CPU memory usage increases a lot while you run yolov5Detect.py, you are probably using only the CPU.

`$ nvidia-smi` (if the GPU memory usage does not change while you run yolov5Detect.py, you are probably using only the CPU)

- Inside the container:

Choose some videos to test the YOLO network
Run a test:

```
$ python3 yolov5Detect.py --source
../../yolov5/1_2019-04-04_23-00-00.mp4 --weights
../../yolov5/weights/yolov5l_best_50.pt --conf-thres
0.1 --view-img --nosave --yml teste.yml --yolov5_path
../../yolov5
```

ps: check all the filenames. The command above is an example.

```
$ xhost -local:docker
```

6. Tracking tool

- For tracking the ants, we run our implementation of the RRANSAC, which reads the .yml files generated by the yoloVideoDetect.py and creates a new .yml file with the tracks. For more information see the Readme.md file on /labcog/contador/dataRRANSAC/.

```
Options:
- data: .yml input file
- out: .yml output file with the generated tracks
- outfw: .yml output file with the generated tracks
separated by vectors (frames, ids, position)
- display: shows the video while it runs (optional - it takes
more time to run)
- RRANSAC params: (must have)
  -M          (int) Number of stored models
  -U,         (float) Merge threshold
  -tau_rho    (float) Good model quality threshold
  -tau_T,     (int) Minimum number of time steps needed for a
good model
  -Nw,        (int) Measurement window size
  -ell,       (int) Number of RANSAC iterations
  -tauR,      (int) Inlier threshold (multiplier)
  -Qm,        (float) Q multiplier
  -Rmx,       (float) Rx multiplier
  -Rmy        (float) Ry multiplier


$ ./dataRRANSAC -data exemploInput.yml -M 50 -U 5 -tau_rho
0.1 -tau_T 3 -Nw 100 -ell 200 -tauR 2 -Qm 2 -Rmx 5 -Rmy 17
-display PATH_TO_VIDEO -out exemploInput_RRANSAC.yml
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

All these parameters were manually adjusted. They change according to the YOLO model quality.

7. Counting tool

- For counting the ants, we run the COUNTLINE, which reads the .yml files generated by the dataRRANSAC and creates a .txt file containing all the counts. For more information see the Readme.md file on /labcog/contador/dataCOUNTLINE/.

```
$ python3 dataCountline.py -yml {}  -cline 0 239 640 241 > {^_RRANSAC.yml}_count.txt'
```

PS: -cline defines where the line is positioned on the video frame. The counts occur when the object crosses that line.

8. Analyzing your data:

a. Organizing the data

- If everything is working well, you should proceed to analyze all your data. However, it is crucial to organize all the data and programs well. We organized all the output files for each step (video data, detection, tracking, and counting) on different directories. Also, as we had more than 2000 videos to analyze, we divided them into "days," allowing us to run all the steps for every day recorded. That is a good option because all the steps will run faster, and it makes it easier to find errors (basically, if the number of output files is different between the directories, an error occurred, and you should run that step for the missing file again).
- Do not move the programs from their original paths! Many programs used here access other program files to run, and if you change the directories, they will not find them, returning errors.
- Example of organization:
    - Directory Day_1:

- DATA: 72 video files (.mp4)
- YOLO: 72 .yml files
- RRANSAC: 72 _RRANSAC.yml files
- COUNTS: 72 .txt files + Day1allcounts.txt file (will be discussed later)

9. Running Object Detection Model for all videos:

- You will use a cross-platform command-line tool for executing jobs in parallel to run the detection for all the videos (as well as for the tracking and counting steps) called RUSH.
- Install RUSH: https://github.com/shenwei356/rush
- You must run this command for every directory containing the video data. That step will take much time according to the number of files that will be analyzed and the computer settings.

```
# 1. run the YOLO detector
find /PATH_TO_DIRECTORY_CONTAINING_THE_VIDEOS_FROM_THAT_DAY
-name "*.mp4" | rush "python3 yolov5Detect.py --source
../../yolov5/1_2019-04-04_23-00-00.mp4 --weights
../../yolov5/weights/yolov5l_best_50.pt --conf-thres 0.1
--view-img --nosave --yolov5_path ../../yolov5 -out
{^.mp4}.yml"
```

10. Running Tracking tool for all videos:

- You must run this command for every directory containing the YOLO detections.

```
# 2. run dataRRANSAC
find
/PATH_TO_DIRECTORY_CONTAINING_THE_DETECTIONS_FROM_THAT_DAY
-name "*.yml" | rush "./dataRRANSAC -data {} -M 50 -U 5
-tau_rho 0.1 -tau_T 3 -Nw 100 -ell 200 -tauR 2 -Qm 2 -Rmx 5
-Rmy 17 -out {^.yml}_RRANSAC.yml"
```

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

11. Running Counting tool for all videos:

- You must run this command for every directory containing the YOLO detections.

```
# 3. run Countline
find
/PATH_TO_DIRECTORY_CONTAINING_THE_TRACKINGS_FROM_THAT_DAY
-name '*_RRANSAC.yml' | rush 'python3 dataCountline.py -yml
{}  -cline 0 239 640 241 > {^_RRANSAC.yml}_count.txt'

cd  /PATH_TO_DIRECTORY_CONTAINING_THE_COUNTS_FROM_THAT_DAY
# 4. join all counts
tail $(find
/PATH_TO_DIRECTORY_CONTAINING_THE_COUNTS_FROM_THAT_DAY -name
"*_count.txt")  | grep -Po "^==>\s+(\S+)|(\d+)$" | tr '\n'
';' | tr -d '=' | tr '>' '\n' >> Day1allcounts.txt
```

- **PS: PAY ATTENTION TO FILE NAMES AND PATHS. You might need to change something from our command lines. Remember that if you do not change the name of the output files, they will be overwritten.**
- **After running these steps for all data, we recommend organizing all the .txt files containing the counts for each day on a single file (.csv, .xml, or .txt)**

12. Some tips and reading suggestions

YOLO:
https://pjreddie.com/darknet/yolo/
https://towardsdatascience.com/yolo-v5-is-here-b668ce2a4908
https://medium.com/deelvin-machine-learning/yolov4-vs-yolov5-db1e0ac7962b
https://blog.roboflow.com/yolov5-improvements-and-evaluation/
DOCKER:
https://www.youtube.com/watch?v=sYr4frA_1d8
https://djangostars.com/blog/what-is-docker-and-how-to-use-it-with-python/
NVIDIA-DRIVER:
https://medium.com/@rosdyanakusuma/how-to-install-nvidia-driver-in-ubuntu-18-04-307c25f73259

Authors: Mila Pamplona Barbosa and Marcelo Arruda de Toledo
Last updated: 27/08/2024

`If something returns an error, do not hesitate to google the error. StackOverflow and other online forums contain many different solutions for almost every error we encountered.`

REFERENCES:

Fang, Y., Guo, X., Chen, K., Zhou, Z., & Ye, Q. (2021). **Accurate and automated detection of surface knots on sawn timbers using YOLO-V5 model.** *BioResources*, *16*(3), 5390–5406. https://doi.org/10.15376/biores.16.3.5390-5406

Goodfellow, I. J., Bengio, Y., Courville, A. (2016). **Deep learning.** MIT Press. www.deeplearningbook.org

Naqa, I. El, & Murphy, M. J. (2015). **Machine Learning in Radiation Oncology.** Machine Learning in Radiation Oncology, 3–11. https://doi.org/10.1007/978-3-319-18305-3

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). **You only look once: Unified, real-time object detection.** Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016–December, 779–788.

Toledo, M. A. (2014). **Aspectos Temporais da organização coletiva do forrageamento em formigas saúvas (*Atta sexdens rubropilosa*).** São Paulo: Instituto de Biociências, Universidade de São Paulo. Dissertação de Mestrado em Fisiologia Geral.

Toledo, M.A.F. (2018). **Orientação espacial e comportamento coletivo em formigas saúvas.** [doi:10.11606/T.41.2018.tde-20092018-110950]. São Paulo: Instituto de Biociências, Universidade de São Paulo. Tese de Doutorado em Fisiologia Geral.

Zhang, X.-D. (2017). **Chapter 6 Machine Learning.** In *A Matrix Algebra Approach to Artificial Intelligence* (Vol. 45, Issue 13). https://books.google.ca/books?id=EoYBngEACAAJ&dq=mitchell+machine+learning+1997&hl=en&sa=X&ved=0ahUKEwiomdqfj8TkAhWGslkKHRCbAtoQ6AEIKjAA

Zhu, X., Lyu, S., Wang, X., and Zhao, Q. (2021). ***TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios.*** 2778–2788. https://doi.org/10.1109/iccvw54120.2021.00312