**README**

This file contains a general description of the technical solution offered by **COPICK** package, a naïve adaptation of OT-2 liquid handler to work as a colony picker.

**1. GENERAL DESCRIPTION**

COPICK hardware assembly includes the following content:

* A custom-made design of a DIY transilluminator, including CAD modelling and electronic assembly.
* Python based scripts to activate / deactivate transilluminator.
* A custom-made CAD models to attach an image system to OT-2.
* Python based scripts to control basic features of the camera.
* Orchestrator script with auxiliary functions to synchronize the workflow typically executed during a colony picking workflow: camera positioning, plate lighting, image processing, colony detection, colony filtering and robot picking.
* Setting up and calibration notes for the showed implementation based on common issues and software features

**2. HARDWARE ASSEMBLY**

Two pieces of hardware were specifically designed to support image acquisition: a high resolution camera attached to OT-2 robotic arm using a 3D-printed scaffold, a DIY transilluminator (to provide a physical location to take pictures under good illumination and perform picking operation) and a PC to execute image processing algorithms and synchronize the whole workflow.

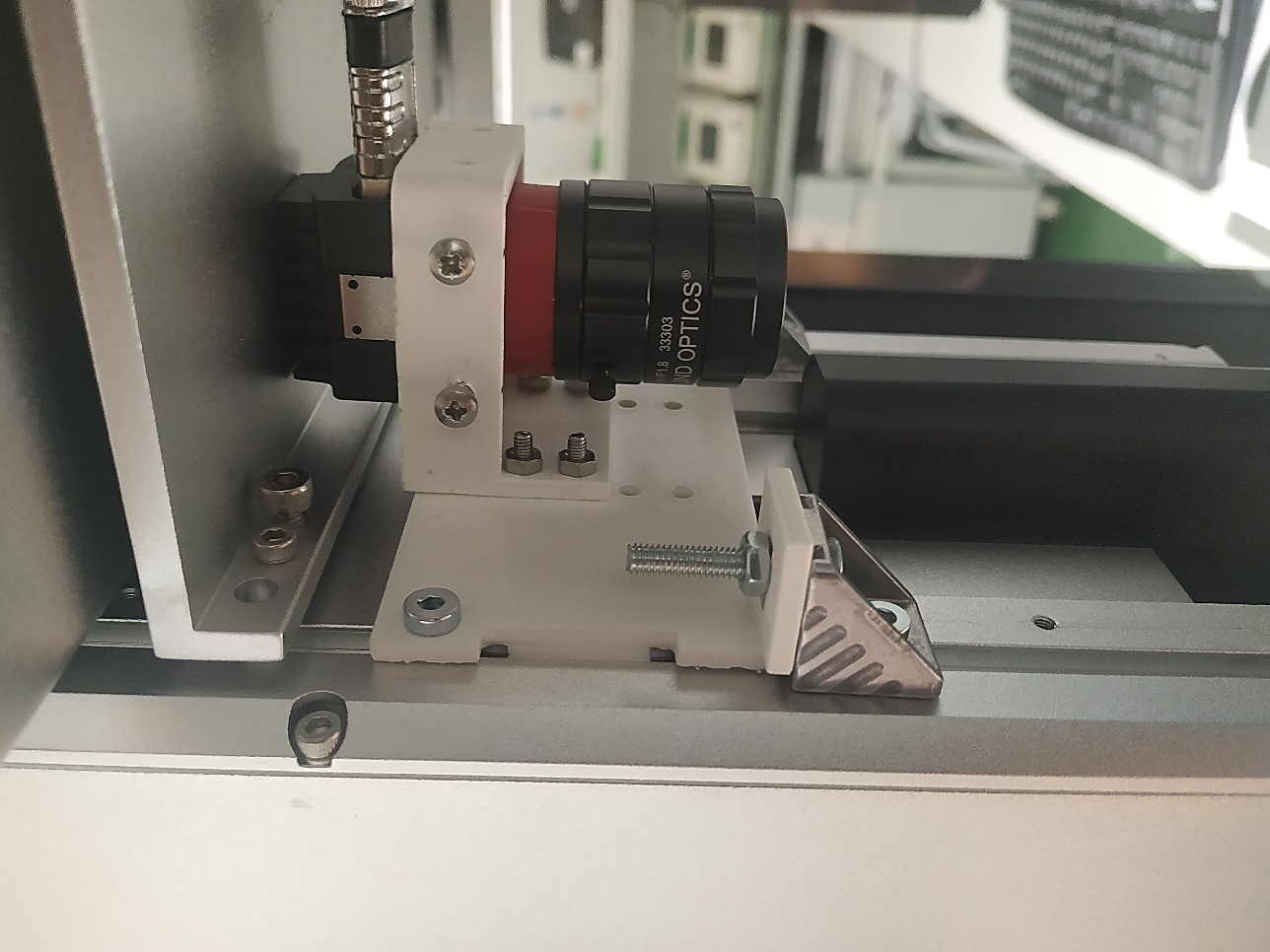
*I) Camera*

Camera scaffold is designed to hold tightly the camera with M3 and M4 screws, as well as Angle Brackets. This structure is also screwed to OT-2 robotic arm using the rails from the back part of the robot, thus no mechanical drilling or gluing is required to attach the camera to OT-2 (see image 1):

Assembly instructions are presented next:

1. Print the provided \*stl files using a rigid material of your choice (i.e. PLA, ABS, etc.). Print the number of copies indicated next:
   * 1. cam\_bracket\_new: 1 units
     2. cam\_adapter\_inv: 1 units

2- Assembly the printed parts as illustrated in the following scheme using M3 and M4 bolts and nuts. M3 bolts and nuts are required for the camera adapter piece. M4 bolts and nuts as well as Angle Bracket pieces are required for the camera bracket piece. You will need to remove the protruding parts of the Angle Bracket pieces with cutting pliers so that the surfaces are completely flat and can be easily aligned to the camera bracket piece and robotic arm.



**Figure 1:** *Camera.mounted on OT-2 robotic arm backpart using 3D-printed scaffold*

1. There are several threads at different heights on robotic arm to screw the 3d printed scaffold. They can be used at will to place the camera if a different model is used. In this implementation we use a USB camera with right angle connector coupled with a 35 mm fixed focus lens. This constrains our camera to be placed in the upper most position.
2. The camera is connected to the PC using a USB 3.0 Micro-B to Type A with locking screws to avoid unexpected disconnections.

1. Additionally, our implementation also offers the possibility to use of a colored glass longpass Filter (500 nm) to allow the detection of GFP when required.

*II) DIY transilluminator*

The presented adaptation of Opentrons machine as a colony picker is based in the use of a light source device inspired in the layout typically found in commercial transilluminators (Figure 2). The purpose of such device is double:

1. Create a working area that fits in the OT-2 deck, where agar plates will be placed to perform required operations
2. Provide a source of vertical light to increase contrast between agar surface and colonies to target

The transilluminator design (Figure 3) is formed by a physical structure and an electronic circuitry in charge of control the emission of light.

*3D Design*

The structure of the DIY transilluminator is modular, and composed by several parts. It contains a base (transillum\_base\_OT2\_v1) designed to occupy two Opentrons slots. Two pieces with shape of an open box (transillum\_box\_walls\_v1) are used to create the vertical space required to place the electronic controller (down) and light source (up). The design of such box lacks of most of the walls to allow the user connect the wires more easily. Both box pieces are connected using a separator piece (transillum\_separator\_v1). A diffusor holder part is placed in the upper part of the assembly to host the diffusor plate. Finally, a light diffusor is used as a flat surface to place your plates. The provided design allows to place Petri dishes and SBS like plates. The dimensions are 134x112x5 mm ± printer tolerance to make it fit in the 3D printed diffusor part.

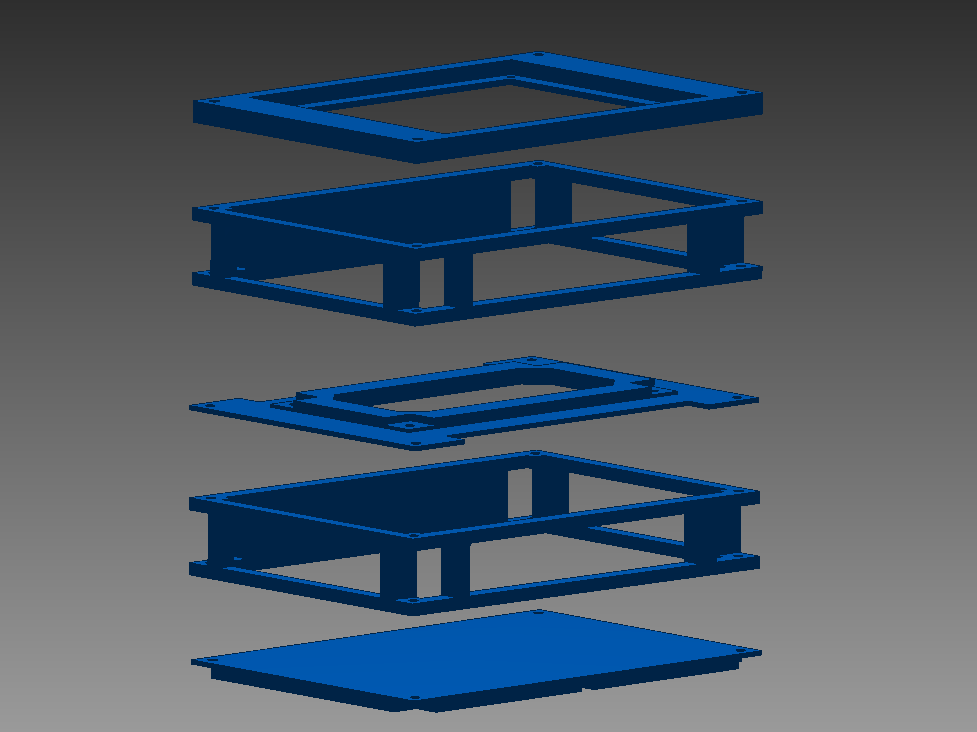


**Figure 2:** Transilluminator

The light diffusor also blurs the light emitted by the leds to avoid a spotted background in the image. You can use whatever plastic plate of your choice, provided that it generates a good diffusion of the light. You should look for a choice that delivers a tradeoff between obtained light diffusion and a proper transmitted light intensity to obtain a good image. In our design, we selected a 5 mm translucid Polyethylene plate purchased in a regular plastic shop (generic).

Assembly instructions are detailed next:

1. Print the provided \*stl files using a rigid material of your choice (i.e. PLA, ABS, etc.). Print the number of copies indicated next:
   * + 1. transillum\_base\_OT2\_v1: 1 units
       2. transillum\_box\_walls\_v1: 2 unit
       3. transillum\_separator\_v1: 1 unit
       4. transillum\_diffusor\_holder\_v1: 1 unit
2. Assembly the printed parts as illustrated in Figure 3 using M3 bolts and nuts. Follow this order:
3. Assemble transillum\_base\_OT2\_v1 and 1 unit of transillum\_box\_walls\_v1.
4. Place the Raspberry Pi coupled with the LED matrix hat inside the assembly. Connect power supply wires and /or Ethernet cable.
5. Assemble transillum\_separator\_v1. Place M3 bolts in the inner part with the head in the upper part. Make sure that these bolts have the head as flat as possible, and that the material is attracted by magnets.
6. Place LED Matrix with M3 magnets on the head of the screws, connecting the panel with the Raspberry Hat as described by manufacturer. Check that the panel is positioned in the center.
7. Assemble one unit of transillum\_box\_walls\_v1 and transillum\_diffusor\_holder\_v1.
8. Finally cover the upper surface with the chosen diffusor.



**Figure 3**: 3D-printed scaffold assembly scheme of the transilluminator

*Electronic circuitry*

The provided design uses an RGB led panel assembled to a Raspberry pi card using a commercial hat that directly connect both devices. The assembly is fed using two power supplies, one for LED shield (4A), and other to feed Raspberry pi (2A). Raspberry pi is connected to PC using either an Ethernet wire or via WiFi.

The assembly instructions, and setting up of the device can be found in the web of the manufacturer:

<https://learn.adafruit.com/adafruit-rgb-matrix-plus-real-time-clock-hat-for-raspberry-pi>

Note that the supplier requires a minimal soldering step to connect a few connectors into the hat adaptor, thus you will need a solder to perform it.

*III) Orchestrator PC*

As mentioned before, colony picking requires the implementation of a computer vision system. This involves executing in a synchronized way different tasks performed by different pieces of hardware. OT-2 is powered by a raspberry Pi, a Linux based automata with a limited computational power. In practice this means that it should be used mainly dedicated to control the physical movement of robot, and thus, excluding it as a resource to execute image processing algorithms.

This implementation has chosen as a solution to embed a PC to manage image processing and at the same time control robot movement in a synchronized fashion using an orchestrator script. Details about software implementation are given in further sections

**3. SOFTWARE**

This solution is fully developed on Python. It contains several scripts to operate the required devices for colony picking: transilluminator, camera, orchestrator PC and OT-2. They are described next.

*I) Transilluminator*

The commercial Raspberry Pi used in the transilluminator assembly comes without any OS on board. This means that you have to install it into an SD card to make your Pi start. Instructions to do this can be found in Raspberry official webpage:

<https://www.raspberrypi.com/documentation/computers/getting-started.html#installing-the-operating-system>

Adafruit LED panel and hat adaptor are actioned in this design using their own Python library (instructions about how to install them are in their webpage: <https://learn.adafruit.com/adafruit-rgb-matrix-plus-real-time-clock-hat-for-raspberry-pi/driving-matrices>), thus be sure that your OS has a Python compiler preinstalled.

The logic of operation is straightforward: the scripts provided within this package connect and disconnect the led panel a fixed number of seconds (5, 15 or 60) when executed. The orchestrator software executed in the PC login to the Raspberry Pi via SSH, and execute the desired script using SSH bash commands. Then the connection is closed. Thus you are required to enable SSH on boot to make your Raspberry Pi accessible using this mode each time you switch on. There are many tutorials that explains how to do this. We write here one of them as an illustration:

<https://phoenixnap.com/kb/enable-ssh-raspberry-pi>

To ensure a consistent script execution, you need to set a static IP into your Raspberry Pi. Again, there are many tutorials about how to do this. Here we provide one of them:

<https://www.luisllamas.es/raspberry-pi-ip-estatica/>

Provided scripts allow to activate illumination in white and blue light to obtain colony footprint and gather GFP based images. In the provided software, these scripts have to be stored in the following path of your Raspberry Pi:

*\rpi-rgb-led-matrix\bindings \python\samples\*

Finally, the Opentrons definition of the transilluminator setup is also provided (named transillum\_omnitray\_v2). The definition of the software was performed using a much larger height than actually is, but this is handled via code by applying displacements in z axis when picking colonies. You will have to adjust such z offsets depending on your robot hardware, Petri Dish / Omnitray model and ml of agar poured. It is commented in the OT-2 configuration document.

*II) Camera*

The camera used in this solution is an Allied Vision Alvium (model: 1800 U-500c, 1/2.5" 5.0MP C-Mount, Right Angle USB 3.1 Color Camera). The manufacturer offers a GUI to manually control the camera, but also Python library to control the hardware directly. Both options are included in a software development kit called ViMBA, which is readily downloadable from the official webpage:

<https://www.alliedvision.com/en/support/software-downloads/>

The implemented script make use of several functions of this package in charge of adjusting capturing parameters such as the gain, the image format or the storage path, thus it is required to install it before any use.

*III) Orchestrator PC*

Required libraries

The following libraries are used by orchestrator script. Be sure to download them before setting up:

* Paramiko
* scp
* time
* datetime
* vimba
* torch
* torchvision
* detectron2\*
* numpy
* cv2
* matplotlib
* copy
* csv
* skimage

\*Detectron2 is a heavy library with many details to spot when installing and setting up. It is the core of the image analysis script developed in this solution. A full tutorial with details about what is it and how to configure it is included in the “detectron2” folder included in this repository. We direct the reader to address it to avoid installation issues.

Overall workflow

The orchestrator PC execute a python script that, by using SSH connections to different hardware (using a shared network), allows the execution of all the tasks of the following workflow:

1. Robot homing and camera positioning
2. Light activation
3. Image acquisition
4. Image analysis
5. Colony selection and positioning
6. Physical picking using OT-2

SSH is a convenient and reliable way to connect OT-2 and DIY transilluminator, as both of them are powered and controlled by a Raspberry Pi. The script make use of Paramiko and scp libraries to perform SSH connections, execute bash commands on remote equipments and send or update files creating SCPClients.

In order to do this, you should have at hand some data related to SSH configuration with the devices to setting up in this solution:

* IP address, user name and password of transilluminator Raspberry Pi
* IP address, user name and path of public key to stablish an ssh connection with OT-2. You can find a tutorial about how to configure it in official Opentrons webpage:

<https://support.opentrons.com/s/article/Setting-up-SSH-access-to-your-OT-2>

<https://support.opentrons.com/s/article/Copying-files-to-and-from-your-OT-2-with-SCP>

OT-2 – PC synchronization

Colony picking involve the sequential operation of Opentrons and image analysis software to gather the image and infer colony positions. As the camera and image analysis software are out of OT-2’s Raspberry Pi, it is necessary to alternate the control of the workflow between PC and OT-2 robot during execution. Concretely:

1. Orchestrator PC must order OT-2 to position camera above transilluminator
2. OT-2 must wait for imaging process is finished and picking positions are received
3. Simultaneously, orchestrator PC must gather and analyze image. Position of colonies in robot coordinates must be transferred to OT-2 to be accessible by the robot
4. Once coordinates are available, start picking on agar plate.

At the date of release of this software, Opentrons (OT-2) only have two operation modes: using OT-2 GUI app or using Jupyter notebook. This implementation make use of the second option, as it allows this kind of interruptions during execution.

The synchronization is performed by using a simple but reliable approach: a file called “isready.txt” stored in OT-2 Jupyter path is used as asynchronous flag for both machines. While the file contains the word “NO”, the OT-2 did not continue the protocol. When the content changes to “YES”, the robot will access a default csv file in which the colony coordinates will be dumped and will proceed to pick them.

Workflow step details

1. PC orchestrator execution: The script first connects the PC to OT-2 via SSH and start the execution of the Jupyter notebook by bash commands.
2. Robot homing and camera positions: PC waits to OT-2 Jupyter notebook to initiate, home the robot and position the robotic arm above the transilluminator. This period of time was empirically measured and set in the code.
3. White lights on in transilluminator: PC connects to Raspberry Pi of transilluminator device and send the execution bash command of the script “white\_light\_10s”. The Pi execute the Python script and activates the white light.
4. Acquisition of image in bright field: PC execute the image acquisition function and store the image gathered by the camera in memory and in a \*.jpg file.
5. Optional: Blue light and gfp image acquisition: If the attribute “is\_gfp” is True, a second instruction is sent via SSH to transilluminator to execute the script “blue\_light\_10s”. Blue light is activated and a second image is gathered by camera.
6. Colony detection using Detectron2: white image is evaluated in the pretrained neural network based on Detectron2 architecture using the overlapped prediction of seven different set of weights coming from independent trainings. This generate a more conservative prediction of detected colonies, avoiding spurious particularities of using just a unique prediction. The output is a panoptic prediction that is stored in a \*.jpg image containing the footprint of each colony detected in the plate.
7. Colony selection: The initial list of colonies detected by Detectron2 is filtered based in several user criteria: size or color. If gfp detection was activated, fluorescent intensity can be selected as well.
8. Colony position calculation: Based on panoptic prediction and used filter, the selected colonies report a list of XY coordinates in pixel units that can be further shifted to robot coordinates to be effectively picked. This conversion is stored into a csv file (“colony\_list\_plate\_0.csv”).
9. Colony coordinate uploading to OT-2 and flag raising: PC connects OT-2 via SSH and upload new csv file and the “isready.txt” sync file, overwriting the ones already present in the robot.
10. OT-2 waking up and colony picking execution: OT-2 robot will access “isready.txt” file in one of the programmed iterations, and once it detects that the file is switched to “YES”, it will open the csv file with the coordinates and load the positions to pick. The rest of the protocol is executed for all the targeted colonies.

**4. CALIBRATION AND SETTING UP**

*Calibration*

The presented technical implementation relay on a set of parameters that required a manual calibration (name in code added in brackets). Any user should check whether the convenience of these values for their own implementations, and modify them properly. They are detailed next:

1. Camera focus (-) and gain (camera\_settings['GAIN\_WHITE'], camera\_settings['GAIN\_GFP']): Depending on camera height and light conditions of the room where the robot is present, focus must be regulated manually to obtain a sharp vision of colonies. Camera gain is expected to have a huge impact in how the background light is captured. Recall that LED panel used in transilluminator is actually actioned by PWM: this means that columns / rows of LEDs are switched on and off with a fixed frequency. Depending on the adjusted gain, camera is expected to capture these fluctuations at different velocities. There are two possible solutions to this:
   1. You can find a fixed value that delivers a steady and homogeneous light in the image (the implemented one). Even though the image has a “bar like” background luminosity, image detection inferring algorithm is flexible enough to detect colonies ignoring such artifacts in image.
   2. Modify the image acquisition code to gather a large number of images, and compute the mean of the values assigned for each pixel in the whole image set. This will eventually generate an average image smoothly illuminated.
2. mm to pixel factor (transillum\_settings['f']). This factor is used to translate distances measured in the image into real distances (in mm).
3. OT-2 tip – camera offset (offset\_camera\_x, offset\_camera\_y): When calling OT-2 to move to the center of the transilluminator, is the robotic pipette what actually moves to that point. However, camera is shifted respect to that position. You need to adjust this shift to position camera pointing to the actual center of the transilluminator. This offset is included in Jupyter notebook in charge of placing the camera to take the snapshot.
4. Pixel coordinates of the labware center defined in OT-2 labware definition into the image acquisition frame (transillum\_settings['CY\_labware'], transillum\_settings['CX\_labware']). This value is used to compute the actual increment between labware center (center of the transilluminator) and colony positions. In order to do that, a simple protocol can be followed:
   1. Put a small post-it approximately in the center of the transilluminator surface
   2. Order OT-2 to pick up a tip with p20 pipette
   3. Order OT-2 to move the tip to the center of the transilluminator
   4. Use any kind of ink (i.e. violet crystal) to wet the surface of the tip
   5. Move down the tip until touching the post-it (and the tip borders have been marked)
   6. Move up the tip and place the camera in the center of the transilluminator
   7. Take a picture
   8. Open the image and measure manually the center position of the mark in the image

*Setting up*

The setting up of this project is showed next. Follow these steps to start working with it:

1. Install Detectron2 package provided in this repository. Use the instructions provided in the folder of the package.
2. Install the rest of required libraries (latest version of them) in Orchestrator PC. This script was validated with Python version 3.7.11.
3. Modify the store paths of the images and libraries at will in the orchestrator script.
4. Print the provided parts with your 3D printer.
5. Assemble the transilluminator part as described.
6. Attach the camera to the adapter and assemble it to OT-2 robotic arm.
7. Connect power supply, ethernet wiring and USB ports of transilluminator and camera.
8. Use windows PowerShell to connect to Raspberry Pi of transilluminator via SSH and store scripts to control switching or LED panel. Run them to check that the electronic of the LED panel work properly.
9. Take a sample Petri dish and place it on the transilluminator surface. Move manually the robotic arm to place it in the center of the transilluminator.
10. Connect camera GUI software and, by connecting the light, adjust manually the focus and gain of the camera until you obtain a good image of the colonies.
11. Recalibrate the pixel position of the labware center in your assembly following the calibration advices given below.
12. Open Jupyter using Opentrons GUI and upload the provided files.
13. Use Jupyter notebook to recalibrate camera position. You should obtain a centered snapshot of the transilluminator.
14. Open Orchestrator script. Load libraries and gather a sample image of a colony plate executing only image gathering part of the code. Check that light is switched on and image is properly acquired.
15. Run colony detection script. Check that cropped image is correct and inference is stored as a new image.
16. Modify colony selection settings and run the function.
17. Run create\_csv function to obtain the coordinates of the colonies.
18. Check SSH connection with OT-2 by uploading the created csv file. Open Jupyter and check that the csv has been updated and match with the one created.
19. Open calibration notebook provided, adjust labware configuration with your preferences and run it. OT-2 should start spotting colonies (no picking). Check potential drifts from actual position of colonies.
20. Manually modify mm to pixel factor and transillum\_settings['CY\_labware'], transillum\_settings['CX\_labware'] to make robot spot colonies properly. Remember to create and upload a new csv with updated coordinates every time you perform any modification in the parameters.
21. Use more plates with different conditions (number, agar medium, color) to explore the possibilities of the framework.

**5. ISSUES AND COMMENTARIES**

*I) Assembly*

We recommend to place the labware in the left side slots of the Opentrons, as it is will be easier for the user to connect and place the power supply and Ethernet wires through the robot. You can also connect OT-2 and transilluminator by using a wireless router acting as a switch, as Raspberry Pi has onboard WiFi connectivity.

*II) Optimizing colony detection*

Although Detectron 2 provides a robust colony prediction, it can also deliver incorrect predictions. Additionally, the provided algorithm makes use of a traditional segmentation algorithm to find the position of the agar plate the image, to gather the crop containing only the agar plate. In some cases, this segmentation can be performed incorrectly. Typical causes are bad plate positioning, bad agar quality (too dry agar), too small colonies, presence of weird objects in agar plate or bad illumination settings. You may want to perform different tests playing with all these parameters to find the conditions in which predictions are robust. Try for example to switch off rail LEDS of OT-2, move the plate or reduce the thickness of the diffusor.

*III) Refining picking coordinates*

After adjusting calibration parameters detailed below, it can be possible that picking may not be fully accurate and require further adjustments. The best approach to do that is to run a picking test modifying manually the picking height to avoid pick the colonies (z\_agar, z\_pick in Jupyter notebook), and manually modify the CX\_labware / CY\_labware parameters. These two parameters act as a XY shift in coordinates of all measurements, so they can be modified to match image predictions with actual spatial position in agar plate. Also have in mind that mm to pixel ratio is a parameter that converts a pixel coordinate (with a value range of 0-2500) to a physical space coordinate (in mm, value range of -45+45). There is a loss of accuracy in such mapping, thus be aware that performed adjustment to this parameter is correct.

**6. CONTACT**

The original content of this package is under MIT license. This means you are free to use, modify or distribute it provided it is not performed with commercial purposes.

Although COPICK package is great, it is quite far from being perfect ;). We are happy to receive any feedback you feel it may be interesting to improve this solution. Please mail to [david.rodriguez@cnb.csic.es](mailto:david.rodriguez@cnb.csic.es)