Image processing for AdaptTest project

A guide and tutorial

Cyril Bozonnet, INRAE (PIAF laboratory, Clermont-Ferrand)

cyril.bozonnet@inrae.fr

Code on Github: https://github.com/cyrilbz/arabido-J

version 2: December 2025

Updates: Segmentation using Deep learning!

It is not needed anymore to train a specific Ilastik model per site. I trained a deeplearning segmentation model at the pot scale using images coming from 4 different sites (100 pots images in the training set + 20 in the test set). The model has been trained using BiaPy, a no-code Python library used to create deeplearning models. The model has then been exported in the BioImageModel Zoo (BMZ) format, so it can be run by DeepImageJ, the ImageJ plugin for deep learning.

A new version of the segmentation and analyze macro has been uploaded on the Github repository (segmentation_analyze_pots_deeplearning.ijm), as well as the BMZ model. In the macro, the call to Ilastik for segmentation at the tray scale has been replaced by a call to DeepImageJ at the pot scale. I added also an overlay of the segmentation results on the original image so one can check the performances after the run. Nothing changes in the way to use the macro (you only need to run the macro, select a folder of registered images when asked, and let it do the job) except that you need to install the model and verify that it works before launching it (see below). The macro takes around 2 minutes per tray (when running on CPU) so for a full data set your computer might be busy for several hours.

Installation procedure for the deeplearning model:

a) Install the DeepImageJ plugin: add DeepImageJ as update-site (“Help>update>Manage update sites” and then “Apply Changes”)

b) Install the model: “Plugins > DeepImageJ>Install model manually” and then browse to the .zip model is located in the bmz_model/ folder in the repository. Click on”Install”

c) Verify the installation: Go to “Plugins > DeepImageJ> DeepImageJ Run” and check that the model appears there and that it works by clicking on “Run on test”.

d) If the model does not appear you can unzip the folder and paste it in Fiji/models/ and then it should now appear when you try step c) again.

e) Run “Plugins > DeepImageJ> Create Macro” and after selecting the correct folder, check that what is written after “model_path” corresponds to what is written line 77 of the ImageJ macro (call to DeepImageJ). It this is the case, you are good to go! If not, correct it.

version 1: Mai 2025

Figure 1: Example of image to be analyzed

Overview

Global analysis of the dataset:

The images from the AdaptTest project are very diverse regarding several criteria:

1. Images resolutions: cameras do not all have the same resolution.
2. Tray orientation: trays are oriented either horizontally or vertically depending on the site and they are never perfectly aligned with each other within a site.
3. Huge diversity in color and texture of leaves & soil within and across sites: due to varying soil type/quality, plant genotypes, changes throughout growth season, non-uniform watering, ...
4. Plant size can be wider than the pot they grow in.

Solution to points 1 &2:

A semi-automatic image registration algorithm.

Created with the help of the image.sc community (https://forum.image.sc/t/registration-images-of-multiple-plant-pots/110341/3) and further adapted to the present cases. Aligning images with respect to a predefined template resolves all uniformities regarding alignment, orientation and resolution.

Solution to point3:

Use of a machine learning algorithm to perform plant segmentation, with one classifier per site.

The power of a well renowned pixel classification tool (Ilastik) helps segmenting plants from background despite colour changes throughout the growth season. It is not perfect as its generalization capability is limited (it is not deep learning) and the images resolution at the plant scale is not huge, but it performs correctly and can be used without prior skills in image processing so that a classifier can be trained easily at each sites.

Solution to point4:

No solution found. The analysis will be carried pot by pot. Hence, any plant that grows larger that its pot will create artifacts in the neighbor pots.

How to carry the pot by pot analysis and extract data:

Through an **ImageJ macro** that takes a stack of image and an Ilastik model & executable locations as inputs, performs plant segmentation followed by morphological operations to clean up the segmentation and then measure plant area within each pot using a moving square as region of interest. Plant area fractions within each pot are written in a .csv file, along with pot index and file original image name (which contains tray number, date, and site ID).

Data analysis:

I wrote a minimal Python program that plots growth curves from the .csv data file. An example in figure 2. Data processing can be carried using any tool of your liking (R, Python, Excel,..). Growth curves are sometimes noisy, which might be due to actual size fluctuations but also color fluctuations that are might be poorly captured by the segmentation model.

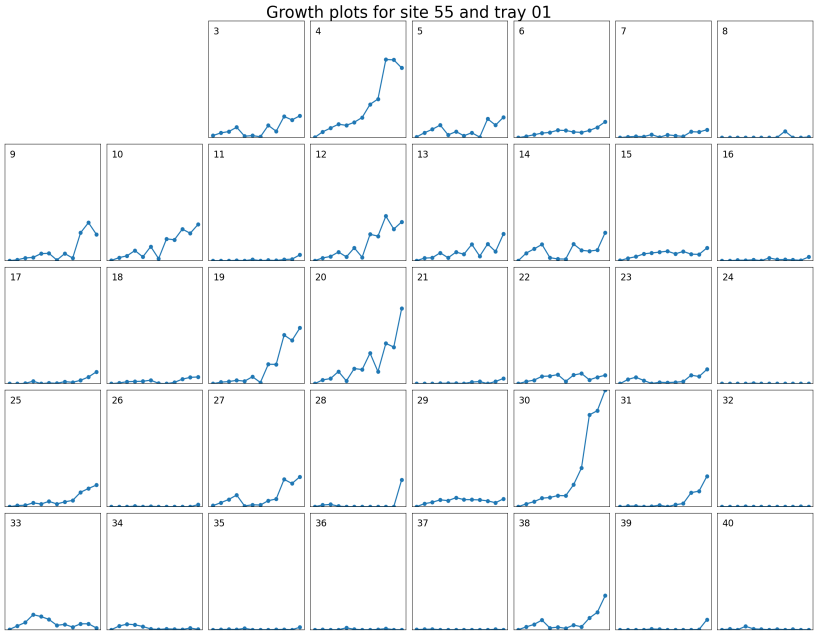


Figure 2: Growth curves obtained using the present methodology

How does it work

Image registration

The ImageJ macro (Registration_trays.ijm) can be run on a folder containing images to register.

The goal is to align 4 specific points in the image regarding a template/reference. When running the script, the template is displayed with the 4 reference points located near the 4 corners of the template image (see right image in figure 3). The template corresponds to a tray oriented horizontally. These 4 reference points are also displayed (not at the same scale) on the image to align (see left image in figure 3).

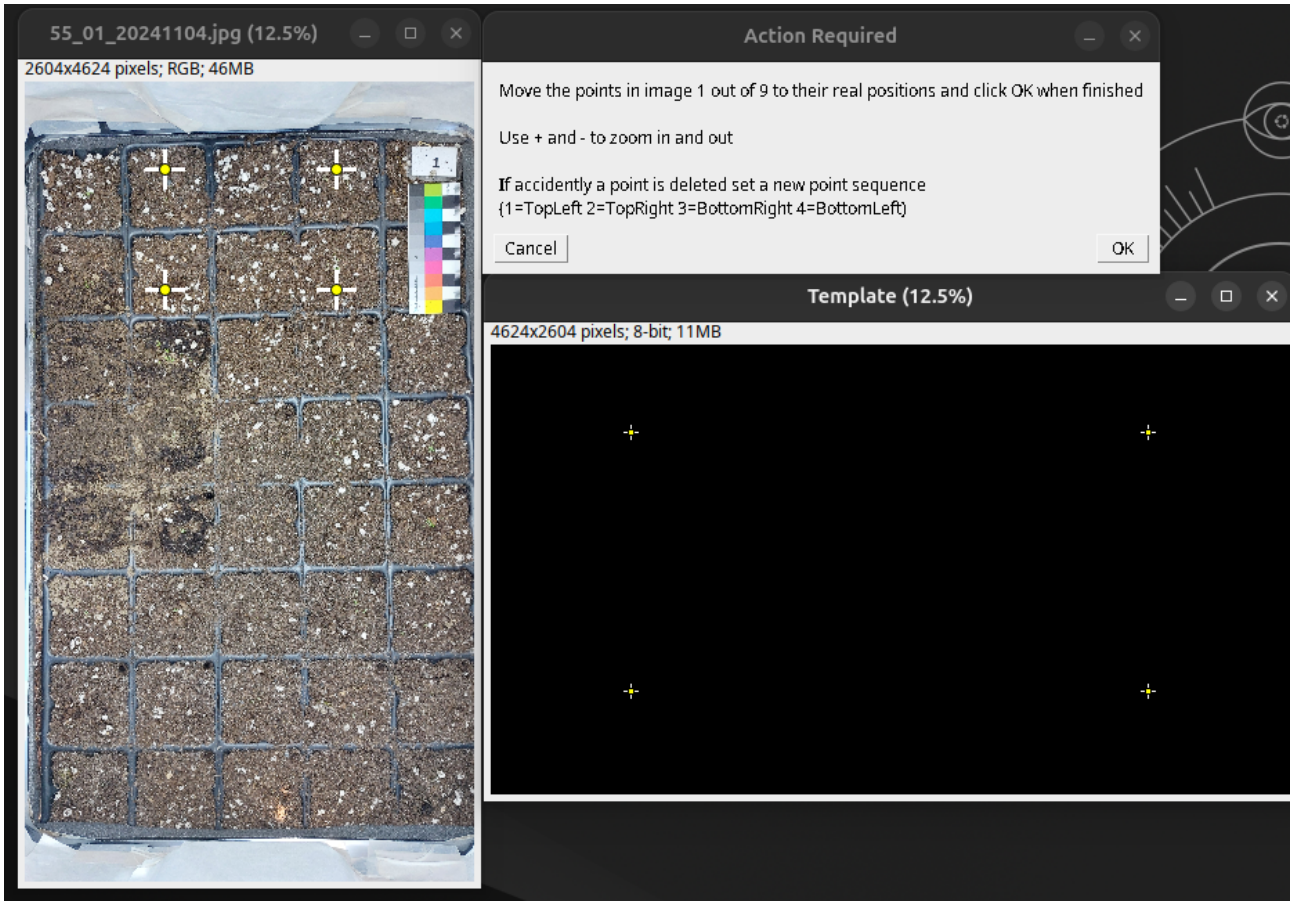


Figure 3: Image registration macro: how it works! Left: Image to align with the 4 initial positions of the reference points. Bottom right: template image with the fours reference points location. Top right: dialog box where the user can click on "OK" once the reference points have been moved to their desired locations.

The user must carefully select (left click), drag and drop the 4 crosses at their desired locations in the image, see middle image in figure 4. For the points to be selected, the user must have the point/multi-point tool selected in ImageJ shape & tool selection (see top right image in figure 4). As an example, the top left cross must go to the location that will become the top left after registration, i.e. the top right of the present image (see middle image in figure 4). The reference locations are always located one pot apart from the sides of the trays to avoid any issues with non-visible tray

corners. Once all 4 points have been moved to their respective final locations, the user can click OK in the text box and the next image in the directory will open.

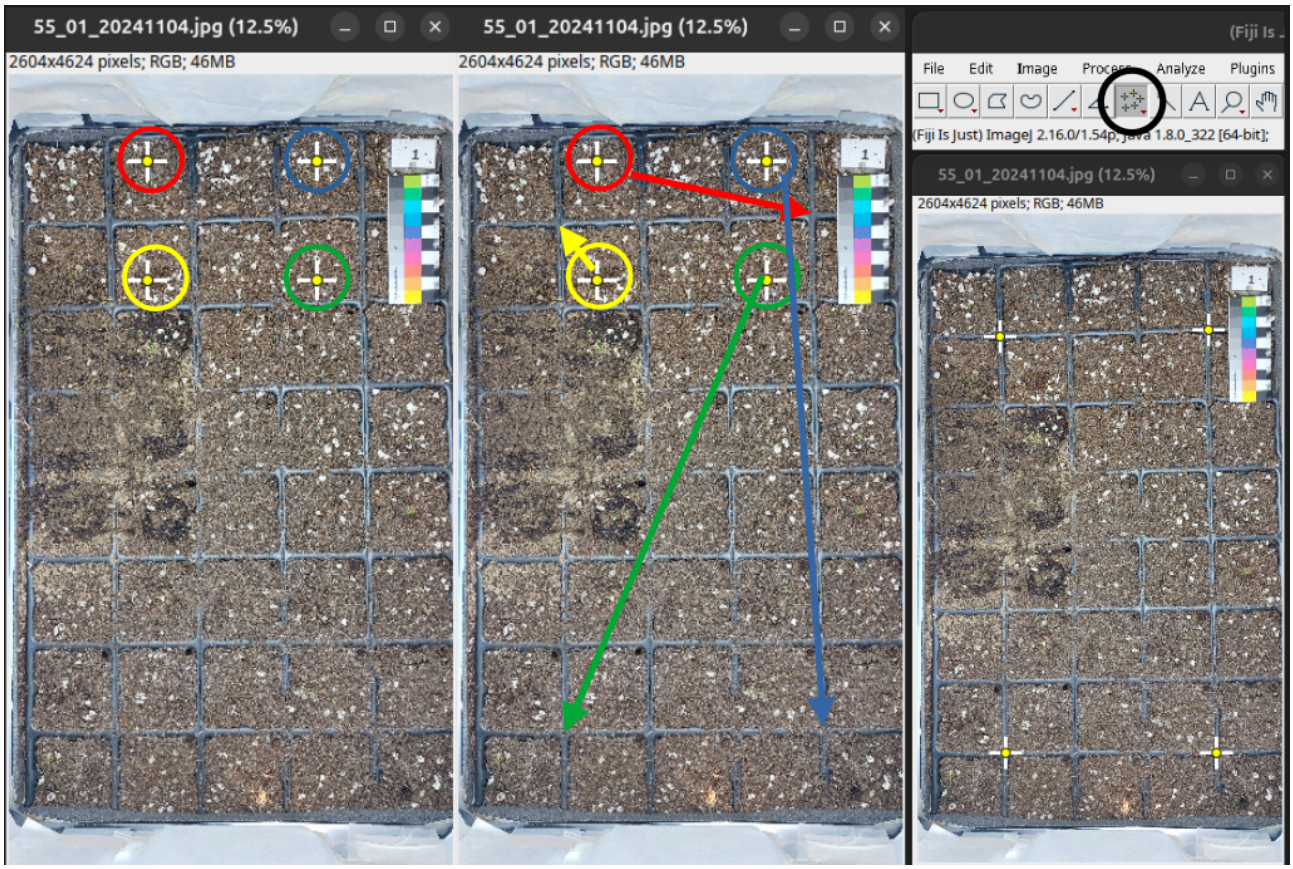


Figure 4: Left: image to align with the 4 reference points highlighted. Middle: the arrows describe the way each point should be moved to its desired location. Right: final point location, and screenshot of ImageJ tool bar to check if the correct tool is selected in case the user cannot select the points.

Once the user has moved the reference points on all images to register, the user is asked to select an output directory (one can be created at that moment using a right-click), then registration takes place and all images get registered. When saving the images, a prefix “r_” is added to the file name.

Ilastik pixel classification model training (deprecated using version 2)

To segment plants from the background, one have to train a specific pixel classifier using Ilastik, a user friendly supervised machine learning tool that allows easy training of pixel classifiers. Installations instructions and tutorial for the pixel classification tool can be found here: https://www.ilastik.org/, https://www.ilastik.org/documentation/basics/installation, https://www.ilastik.org/documentation/pixelclassification/pixelclassification. These links provide enough information and advices to start working on your pixel classification model. **This procedure does not require any prior skills in image processing or machine learning.**

I expand below on some general advices when training a model:

- Data selection: you must train the model on several images representative of the data-set (up to 4 or 5 images) and its variability across the season. No need to use the full images, but you can crop some part of the images to train the model on a small but representative part of the images (using ImageJ, one can use the rectangle selection tool then select the part of interest, then use Ctrl+Maj+X to crop the image and then save it). As an example, here below is the training set I used for the “Clermont” model. During training, the user can change the image to train on using the “Current view” button in the left pannel (“3. Training” section). Do not select too similar images, nor put all your labels in “easy to segment” parts, it makes the model less able to generalize.

Figure 5: Example of training data set for the "Clermont" model.

– Features selection: I advise to select the following features that ignore very small scale variations in the image

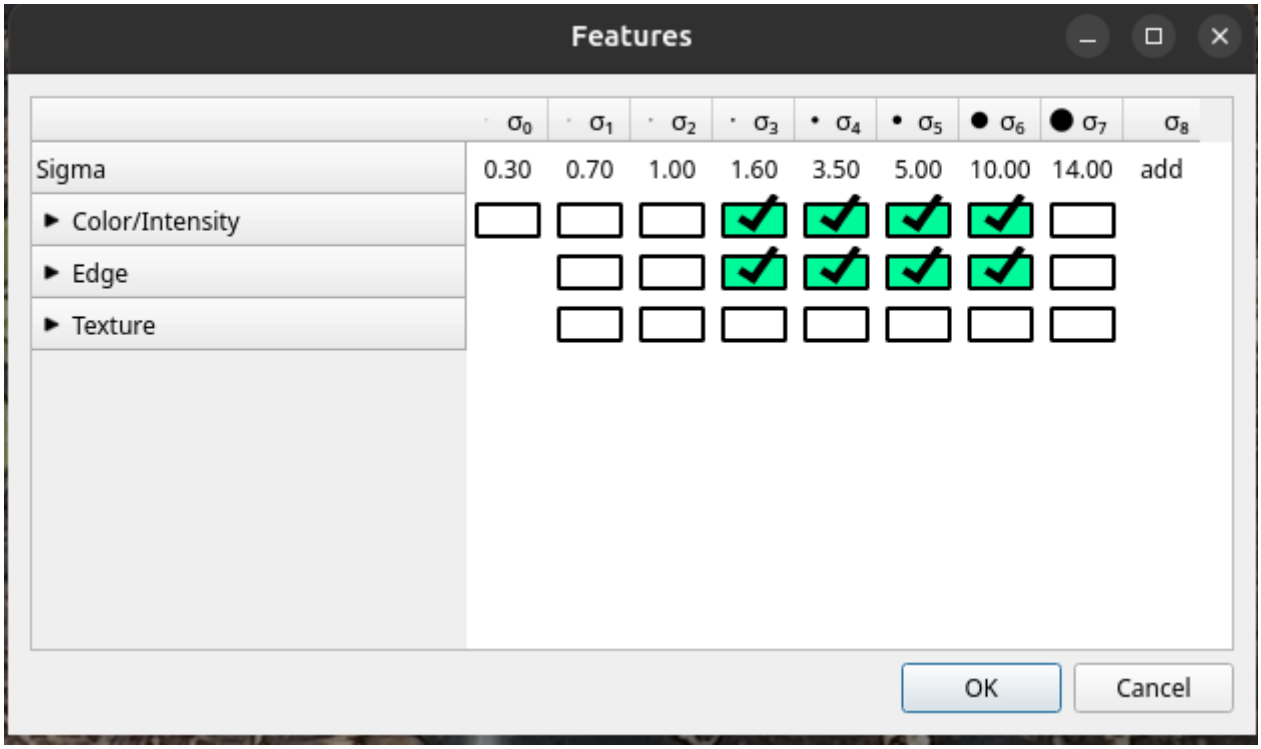
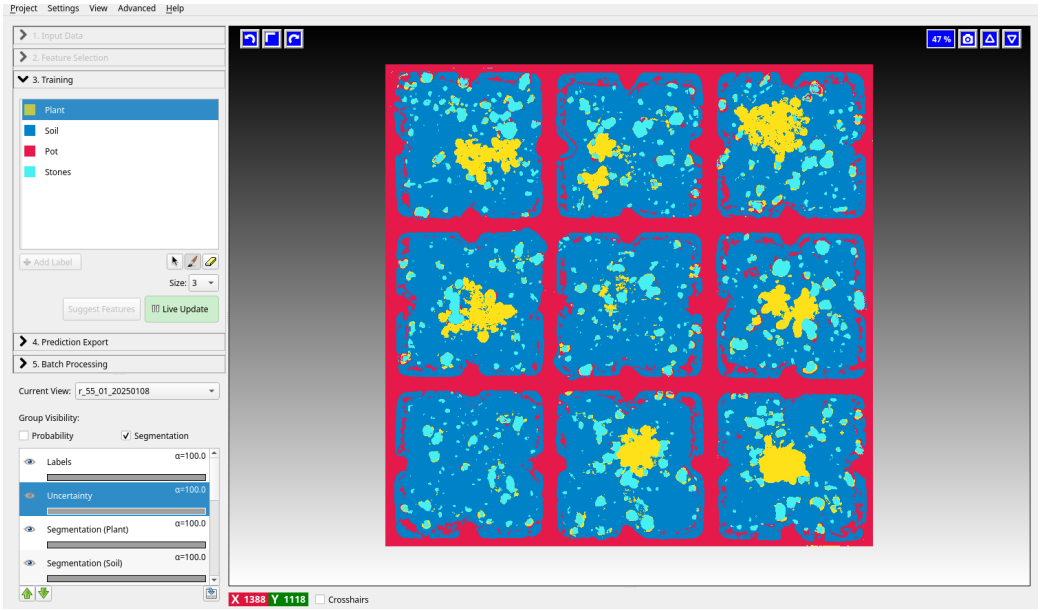


Figure 6: Features to select.

- Label selection: Create as many labels as you think there are different types of objects in the images. **The first label should always be the plant.** Then there must be one label for the soil, and one for the pots. You can then add other labels, if relevant, for stones, wood pieces,…
- Use the “Live Update” button to see the result improving in live when labeling images. To go back to Input data or Feature Selection, the “Live update” must be deselected.
- You can easily view the results in terms of probability or segmentation for all or each labels by selecting them in the “group Visibility” section.
- You can always use the eraser to remove any label drawn by mistake.
- Do not forget to save the project regularly “Project > Save project”.



Segmentation and analysis

Now all images of your data set have been registered and you have trained your own pixel classifier. You can now easily run the dedicated ImageJ macro that will call the pixel classifier you created ("segmentation_analyze_pots.ijm").

When running the macro, the user will first be asked:

- to select the Ilastik model to use (you must have closed Ilastik before running the macro)
- to select the directory in which the images are located (note that you must only have images in this directory)

The macro will then, image per image, run Ilastik for the segmentation, clean its output using morphological operations and size filtering (elements of area <200 pixels squared are removed). For each image the area occupied by the plant ("%area" in the output table) is computed pot by pot and the data are stored in a "Summary" table. The user is then finally asked to select where to save the .csv file containing all results.

You can do the post-processing with the tool you want. As an example, I put on the Github repository a Python code (arabido_analysis.py) that plots growth curve from a .csv file.

Advices

Run the two macros on a small set of images first, to try them before running them on larger data set. I advise to use the registration macro on small data sets as it is a time consuming process (between 4 and 6 hours of work for 200 images, might be lower depending on your ease with the procedure). The segmentation and analyze macro can be run on a full data set (takes ~1h30 for 200 images).

For any questions and requests, send an e-mail to cyril.bozonnet@inrae.fr

Future improvements

The present framework generates good results but is sub-optimal in several ways:

- Registration is time consuming for the user
- Image segmentation is not perfect and requires each site to train their own model

In the future, and if my other projects allow me to do so, I will work on Deep Learning based versions of these tools, to enable fully automated registration and high quality plant segmentation for all sites. Anyone that would like to help or know more can reach me.