Guidelines for image based phenotyping of *A. thaliana*

Leonard Blaschek

April 25, 2021

Why we use image based phenotyping

We perform image based automated phenotyping to measure more parameters of plant growth at a higher temporal resolution than we could feasibly do using manual methods. Not only do image based approaches allow us to measure plants' height, area, colour and overall shape throughout their life cycle, the high quality images themselves can be used to measure additional features or phenotypes later on. As with all image analysis approaches, the quality of the input images is the single most important variable in determining success. In the approach implemented by me, two imaging setups are used, depending on the developmental stage of the *A. thaliana* plants. In this document, I will try to summarise what you need to know about image acquisition, automated phenotyping and data analysis.

Contents

1 Rosette growth **2**

2 Inflorescence growth **2**

3 Image metadata **3**

4 Image exploration in Fiji **4**

5 High-throughput analysis in Python and PlantCV **4**

5.1 Installation 5

5.2 Optimisation in Jupyter notebook 5

5.3 Batch analysis of multiple files 5

5.4 Data analysis 5

1 Rosette growth

From germination until bolting, the plants are best imaged from the top. One big advantage of this is that the pots can be left in the tray, meaning that 15 plants can be analysed from a single picture. As you can see from the example image in figure 1, I acquire the images without labels. Instead, I have labels on the side of the pots (see figure 2). That of course means that you have to be very careful to document the order of pots (genotypes/replicates) in each tray, and orient the trays in the same way in each picture (put a label on one side of the tray to make sure you do not accidentally flip it 180°). You also need to carefully document which picture shows which tray. Lastly, bright markers (any colour that stands out, I use light blue) on the outer edges of the tray allow you to easily crop the image to the area of interest in the analysis. Lastly, taking the pictures always in the same order will save you a lot of time in the analysis (see section 3).

Figure 1: *A. thaliana* plants during their vegetative phase (before bolting of the inflorescence stem), imaged from the top. The image should cover the whole tray and the colour card to the left, both of which should be oriented in parallel with the image borders. Acquisition parameters need to be chosen to avoid over-exposure. Also note the blue stickers on the corners of the tray, which are later used to define the area of interest for the image analysis.

In order to avoid over-exposure, the following acquisition parameters have proven successful in our setup:

- 50 mm objective
- 1/125 s exposure time
- f5.6 aperture
- 800 ISO sensitivity
- white balance set to ”Incandescent”

2 Inflorescence growth

After the plants bolt, we switch to imaging from the front. Now we have to image each plant individually. Make sure that the label on the front of the pot is clearly visible in each image. Everything in the image that is *not* the plant or the colour card, such as labels, sticks, pots, *etc.* should be of a colour that is easy to distinguish from the plant. Because *A. thaliana* can be green (leaves), white (flowers), purple (stressed leaves), yellow/beige/brown (plants in senescence), the ideal colour for foreign objects is blue (see stick in figure 3). Just as with the rosette images, taking the pictures always in the same order will save you a lot of time in the analysis (see section 3).
For this stage, the following acquisition parameters have proven successful in our setup:

- 50 mm objective

Figure 2: An *A. thaliana* plant during its reproductive phase (after bolting of the influorescence stem, but **before** binding the stems to a stick), imaged from the front. The image should cover the whole plant and the colour card to the left, both of which should be oriented in parallel with the image borders. Acquisition parameters need to be chosen to avoid over-exposure. Also note the red label on the front of the pot, used for identification. This label can contain any amount of text, but should be blue or red, to avoid colour similarity to the plant itself.

- 1/30 s exposure time
- f13 aperture
- 2000 ISO sensitivity
- white balance set to "Incandescent"

Figure 3: An *A. thaliana* plant during its reproductive phase (**after** binding the stems to a stick), imaged from the front. Just like the label, the stick should ideally be blue to facilitate separation from the plant itself in the analysis.

3 Image metadata

If we want to automate the image analysis, we need to make sure that each image contains all the metadata we need to assign the output to the correct plant. Imagine for example we take an image of a tray with 15 pots from the top and save that image as `DSC_2487.JPG`. If we input this image into the analysis pipeline, it will output results for each of the 15 plants on the tray, which it will

call `DSC_2487.JPG_1, DSC_2487.JPG_2` and so on. But what genotype is plant `DSC_2487.JPG_1` and how old was the plant when the picture was taken? The easiest way to embed metadata in your image is in the image name. If we for example rename our image `DSC_2487.JPG` to `tray2_day13.JPG`, the results for the first plant of the tray will we called `tray2_day13.JPG_1`, which we can very easily match to the tray layouts you have noted down.

You can rename the pictures by hand, but if you have followed my advice above and took the pictures always in the same order, we can also automate the renaming. The following script works on MacOS and Linux, and renames all `.JPG` files in the same folder. So create one folder with the picture from the top and one folder with the pictures from the front. The script renames the images with the day past germination and the number of each tray or plant. For example, if your plants germinated roughly on the 24th of July 2020, you'd set `germination=20200824`. If you have three full trays of plants, you would set the `end=3` for the pictures taken from above, and `end=45` for the pictures of the individual plants taken from the front.

```
#!/bin/bash
# define the date of germination for DPG (days past germination) calculation
germination="20200824"
# number of the first sequentially numbered trays/plants
start=1
# number of the last sequentially numbered trays/plants
end=3
# rename images by DPG and tray/plant number
a=$start
for i in *.JPG; do
  var1="_"
  DPG=$(( ($(date -r "$i" +%s) - $(date --date=$germination +%s) )assign the starting
    / (60*60*24) ))
  mv -i -- "$i" "$DPG$var1$a.jpg"
  if [ "$a" -lt $end ]
  then
    let a=a+1
  else
    let a=$start
  fi
done
```

On Windows, you could install the bash shell as is described here, to use the above script. Or you use the Python script below

```
## TODO: TRANSLATE BASH SCRIPT TO PYTHON
```

4 Image exploration in Fiji

To get a feeling for the images, open one of them in Fiji and explore how you can segment and measure different parts of the plant. A very useful function for this is `Image` → `Adjust` → `Color Threshold...` (use the LAB colour space). To calibrate any height or area measurements, you can define the pixel size using the ruler on the bottom of the colour card.

5 High-throughput analysis in Python and PlantCV

You can do the complete analysis in Fiji, but depending on the amount of images you have, this might be prohibitively time consuming. The alternative is a programmatic approach that automates same principles of image analysis that we would use in Fiji. PlantCV is a Python library that was specifically designed for plant phenotyping. Getting started with it is more complicated than using Fiji, but once it is set up the reward is a highly reproducible analysis pipeline that needs minimal manual intervention. In the following I will outline the initial steps of working with our images in PlantCV. Beyond this short tutorial, I highly recommend the excellent official documentation of

PlantCV. Lastly, if you are already familiar with PlantCV and want to skip ahead to the analysis pipeline optimised for our imaging setup, head over to my Github page.

5.1 Installation

I will assume that most students reading this document are using Windows 10. But even if not, most of what follows is operating system agnostic and will work (almost) identically on MacOS and Linux.

Install Anaconda `Anaconda` and its slimmed down version `Miniconda` are package and environment managers for Python. It allows you to create virtual environments for specific tasks, which simplifies work in Python immensely. If you do not have it installed already, I recommend `Miniconda`. Simply follow the installation instruction for you operating system. If you are unsure, your operating system is almost definitely the 64-bit variant.

Install PlantCV If you are on Windows, start the `Anaconda Prompt`, on MacOS and Linux simply open a terminal window. To test that conda is properly installed, type:

```
conda --version
```

The prompt should then output the installed conda version, for example `conda 4.9.2`. If this worked as expected, we next need to add the channel containing the PlantCV package to the conda configuration:

```
conda config --add channels conda-forge
```

Now we install PlantCV in a new environment called `plantcv`:

```
conda create -n plantcv plantcv
```

5.2 Optimisation in Jupyter notebook

5.3 Batch analysis of multiple files

5.4 Data analysis